Contents

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 Duration and conditions of warranty for this product may be superseded when the product is integrated into (becomes a part of) other HP products.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

U.S. Government Restricted Rights

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

HP E1313A/E1413C User's Manual
Edition 6

Documentation History

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

Edition 1 . . . . . May 1993
Edition 2 . . . . . September 1993
Edition 3 . . . . . March 1994
Edition 4 . . . . . October 1994
Edition 5 . . . . . February 1996
Edition 6 . . . . . August 1996

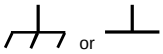
Trademarks

MS-DOS® is a U.S. registered trademark of Microsoft Corporation.
UNIX® is a registered trademark in the United States and other countries, licensed exculsively through X/Open Company Limited.

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.

Declaration of Conformity

according to ISO/IEC Guide 22 and EN 45014

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

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

declares, that the product:

Product Name: 32-Channel or 64-Channel Scanning A/D Converter

Model Number: HP E1313A

Product Options: All

conforms to the following Product Specifications:

Safety: IEC 1010-1 (1990) Incl. Amend 1 (1992)/EN61010-1 (1993)
CSA C22.2 #1010.1 (1992)
UL 1244

EMC: CISPR 11:1990/EN55011 (1991): Group 1, Class A
IEC 801-2:1991/EN50082-1 (1992): 4kV CD
IEC 801-3:1984/EN50082-1 (1992): 3V/m
IEC 801-4:1988/EN50082-1 (1992): 1kV Power Line

Supplementary Information: The product herewith complies with the requirements of the Low Voltage Directive 73/23/EEC and the EMC Directive 89/336/EEC.

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

November 25, 1994 **Jim White, QA Manager**

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

Declaration of Conformity

according to ISO/IEC Guide 22 and EN 45014

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

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

declares, that the product:

Product Name: 64-Channel Scanning A/D Converter

Model Number: HP E1413C

Product Options: All

conforms to the following Product Specifications:

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

EMC: CISPR 11:1990/EN55011 (1991): Group 1 Class A
IEC 801-2:1991/EN50082-1 (1992): 4kV CD, 8kV AD
IEC 801-3:1984/EN50082-1 (1992): 3V/m
IEC 801-4:1988/EN50082-1 (1992): 1kV Power Line
0.5kV Signal Lines

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

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

March 15, 1996 **Jim White, QA Manager**

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

Please fold and tape for mailing

Reader Comment Sheet

HP E1313A/E1413C User's Manual
Edition 6

You can help us improve our manuals by sharing your comments and suggestions. **In appreciation of your time, we will enter you in a quarterly drawing for a Hewlett-Packard Palmtop Personal Computer** (U.S. government employees cannot participate in the drawing).

Your Name	City, State/Province
Company Name	Country
Job Title	Zip/Postal Code
Address	Telephone Number with Area Code

Please list the system controller, operating system, programming language, and plug-in modules you are using.

fold here

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 37 LOVELAND, CO

POSTAGE WILL BE PAID BY ADDRESSEE

HEWLETT-PACKARD COMPANY
Measurement Systems Division
Learning Products Department
P.O. Box 301
Loveland, CO 80539-9984

cut along this line

fold here

Please pencil-in one circle for each statement below:

	Disagree ←				→ Agree
• The documentation is well organized.	O	O	O	O	O
• Instructions are easy to understand.	O	O	O	O	O
• The documentation is clearly written.	O	O	O	O	O
• Examples are clear and useful.	O	O	O	O	O
• Illustrations are clear and helpful.	O	O	O	O	O
• The documentation meets my overall expectations.	O	O	O	O	O

Please write any comments or suggestions below—be specific.

Chapter 1
Getting Started

About This Chapter

Except where noted, all references to the HP E1413 apply to the HP E1313. This chapter will explain hardware configuration before installation in a VXIbus mainframe. By attending to each of these configuration items, your HP Scanning A/D Converter module will not need to be removed from its mainframe later. Chapter contents include:

Configuring the HP Scanning A/D Converter Module

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

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 Scanning A/D Converters use 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 Figures 1-1 and 1-2, and ignore any switch numbering printed on the logical address switch. When installing more than one HP Scanning A/D Converter in a single VXIbus mainframe, set each instrument to a different logical address.

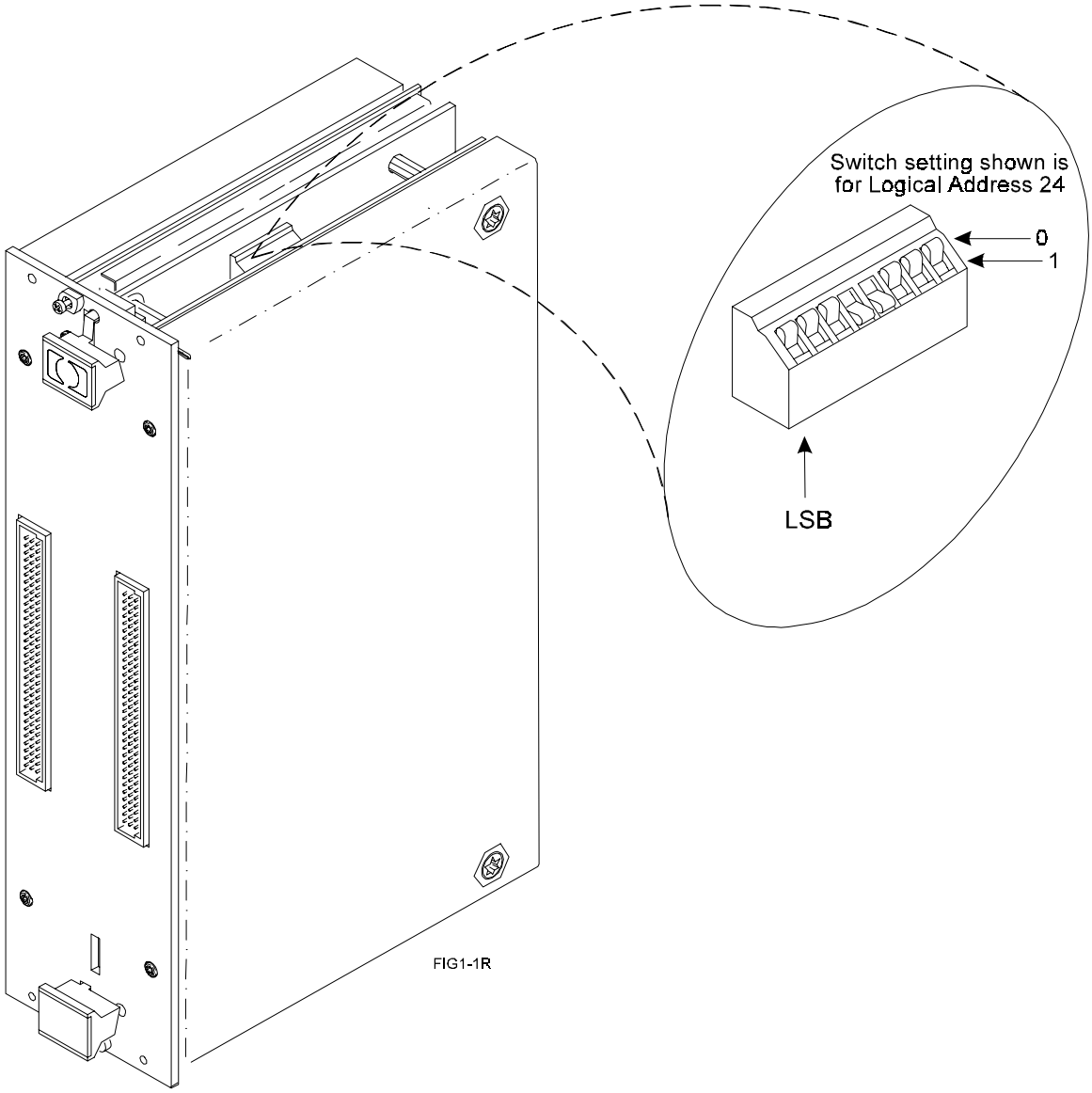


Figure 1-1. Setting the HP E1313 Logical Address

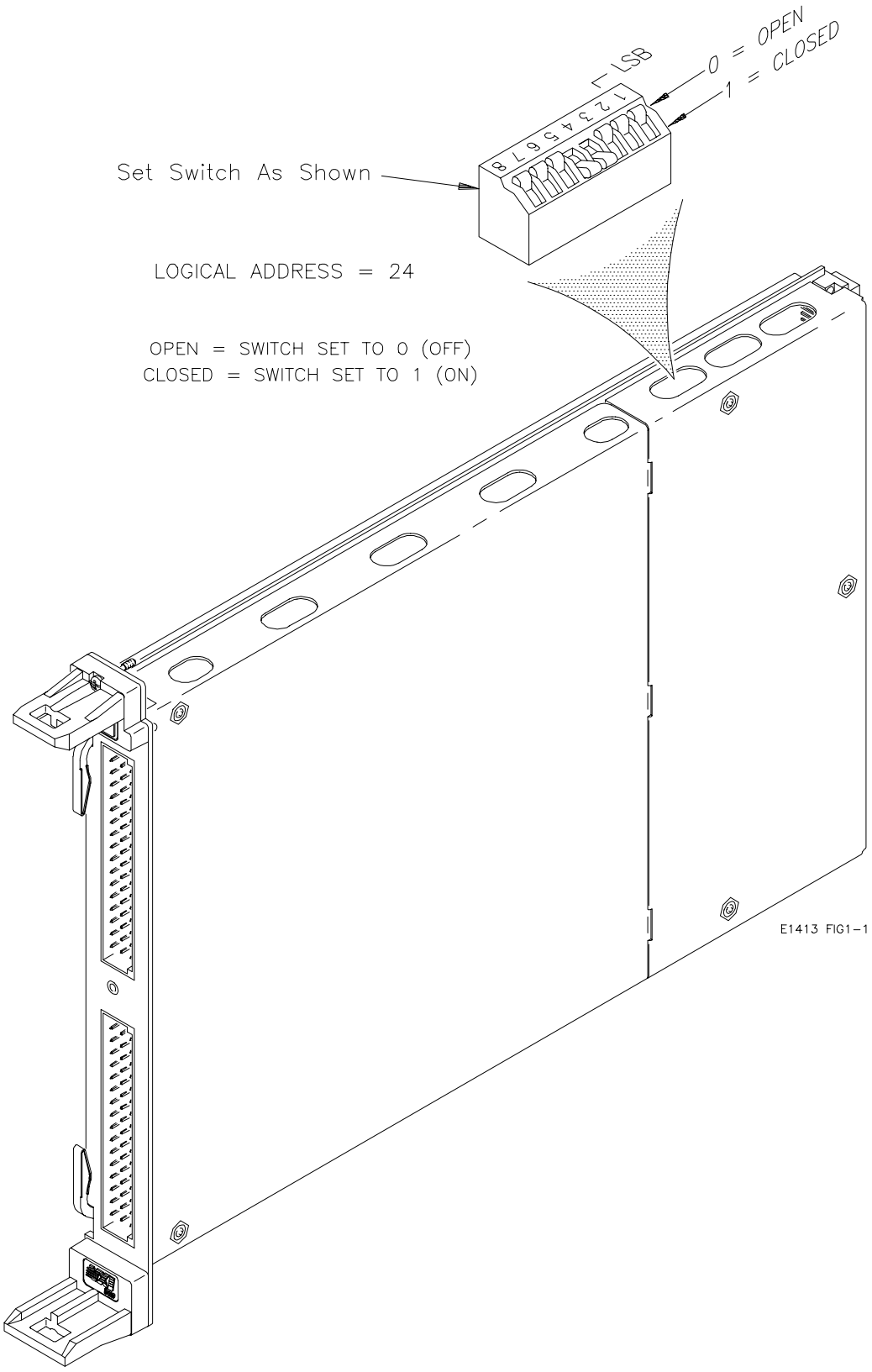


Figure 1-2. Setting the HP E1413 Logical Address

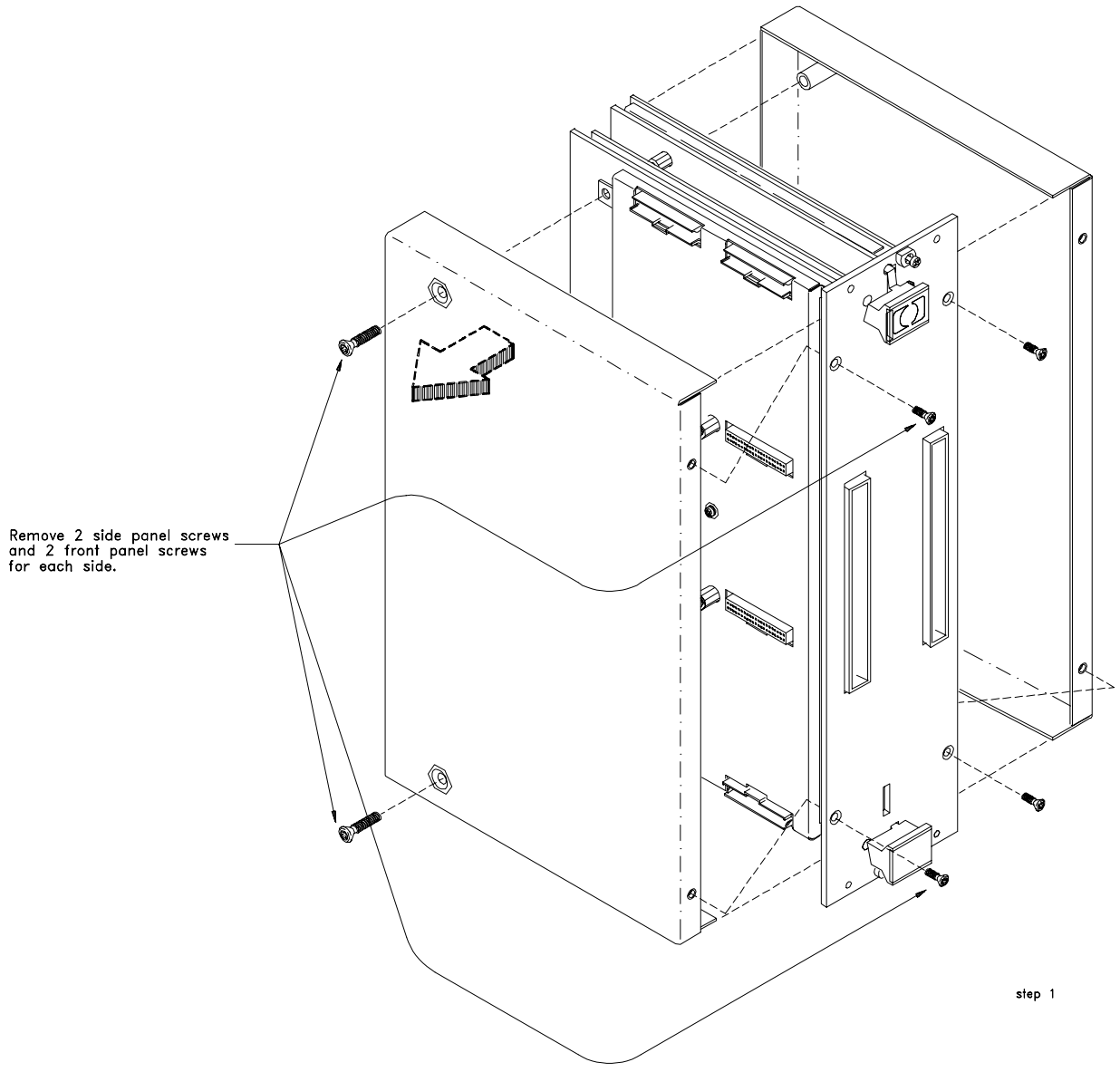
Installing HP E1313 Signal Conditioning Plug-ons

The following illustrations show the steps you will use to install the HP E1313 Signal Conditioning Plug-on modules (SCPs).

CAUTION Use approved Static Discharge Safe handling procedures anytime you have the covers removed from the HP Scanning A/D Converter modules or are handling SCPs.

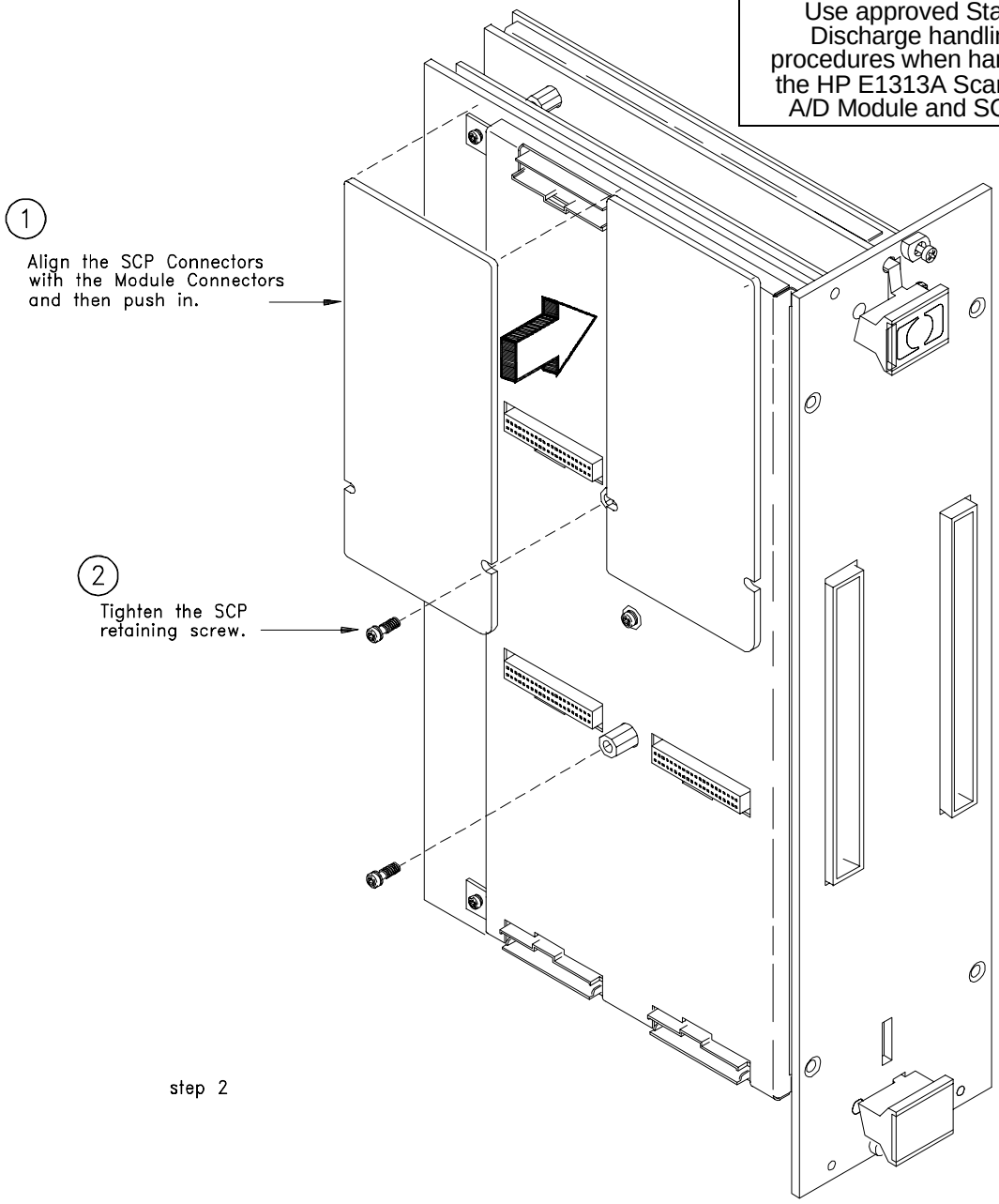
HP E1313 Step 1: Installing SCPs

Remove the cover(s) from the HP E1313



HP E1313 Step 2: Installing SCPs

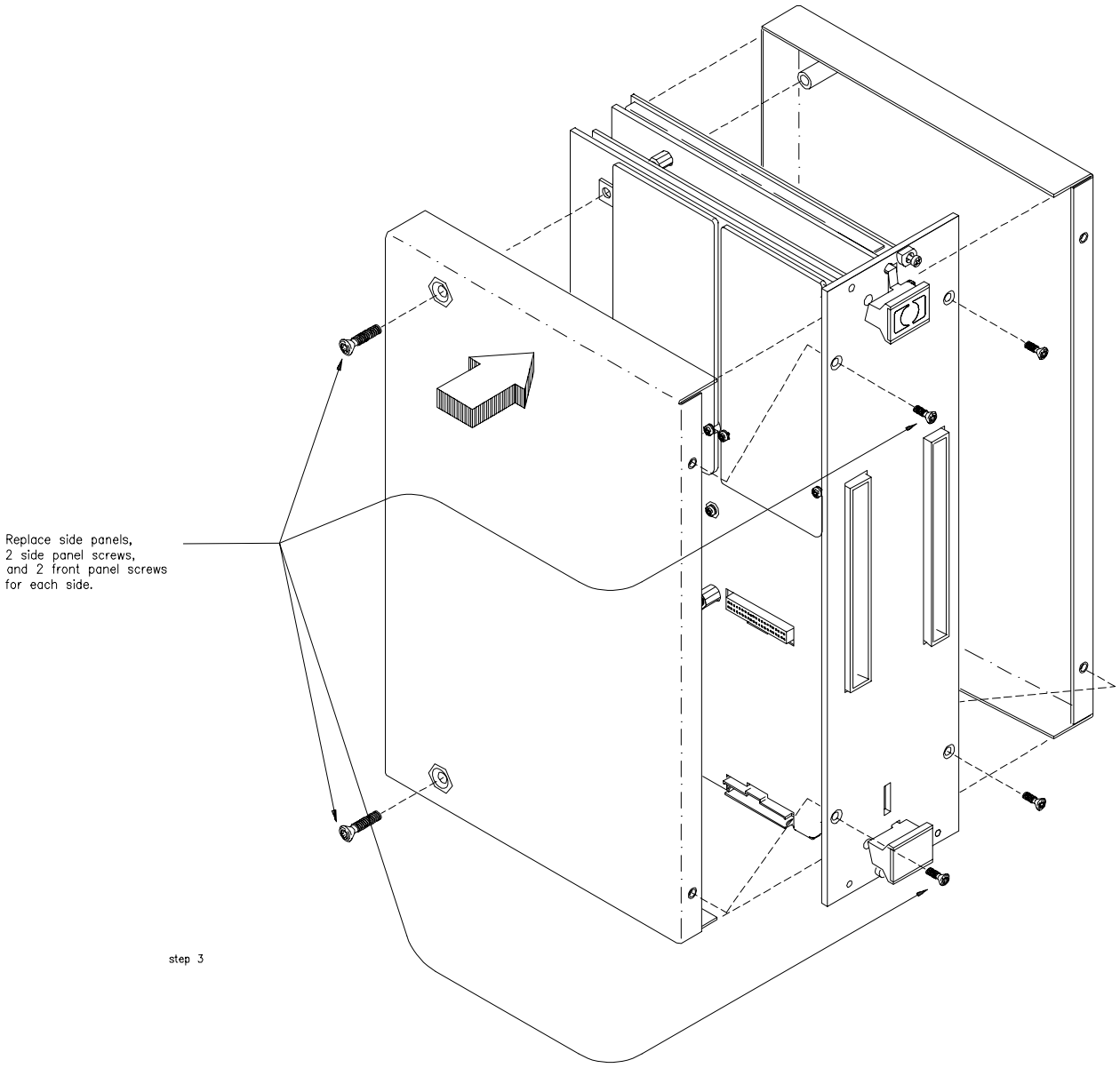
CAUTION
Use approved Static Discharge handling procedures when handling the HP E1313A Scanning A/D Module and SCPs.



NOTE: No more than four HP E1510A Sample and Hold (Option 020) SCPs can be used with an individual HP E1313A.

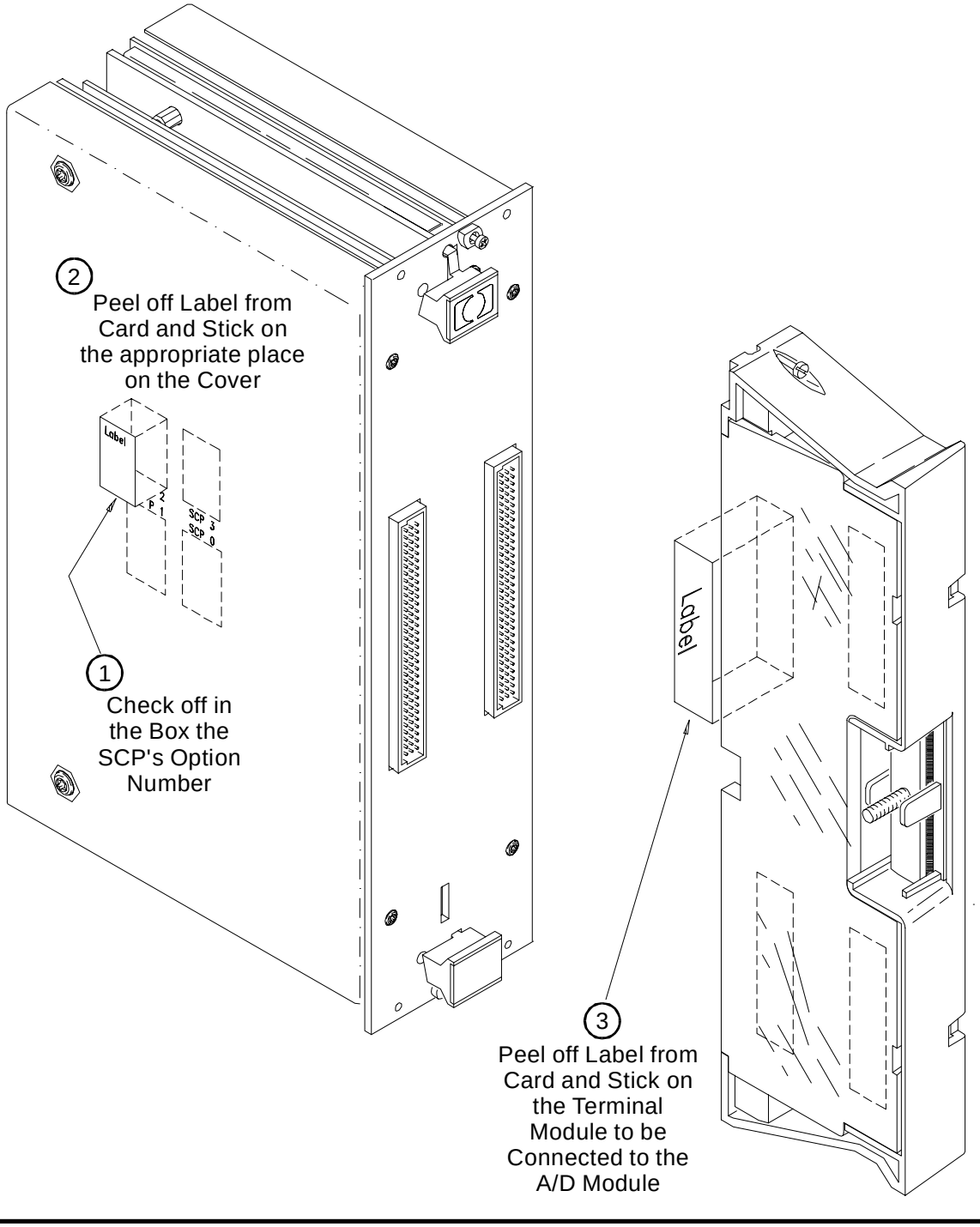
HP E1313 Step 3: Installing SCPs

Reinstall the cover(s) on the HP E1313



HP E1313 Step 4: Installing SCPs

Apply label to the cover of the HP E1313



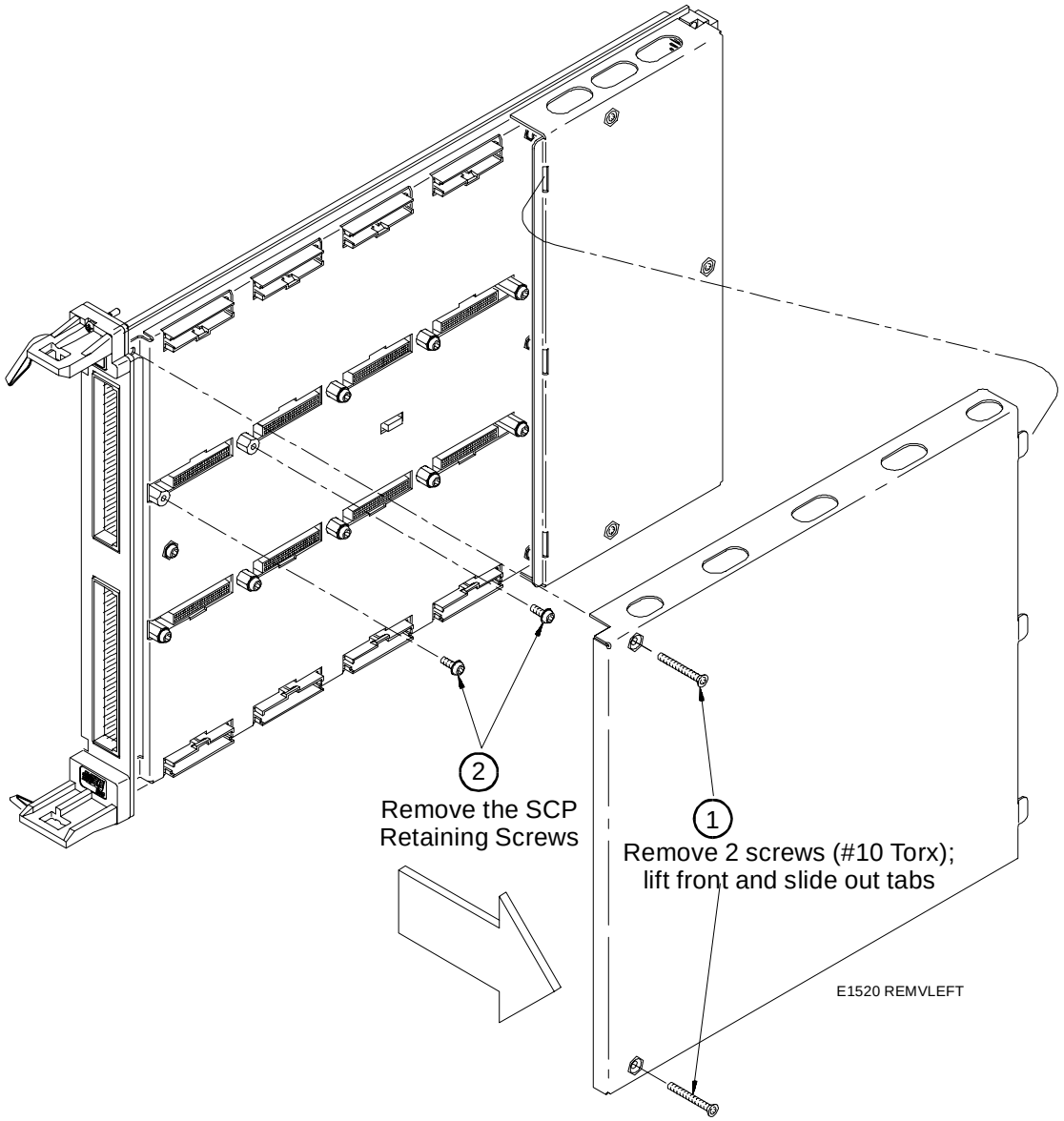
Installing HP E1413 Signal Conditioning Plug-ons

The following illustrations show the steps you will use to install the HP E1413 Signal Conditioning Plug-on modules (SCPs).

CAUTION Use approved Static Discharge Safe handling procedures anytime you have the covers removed from the HP Scanning A/D Converter modules or are handling SCPs.

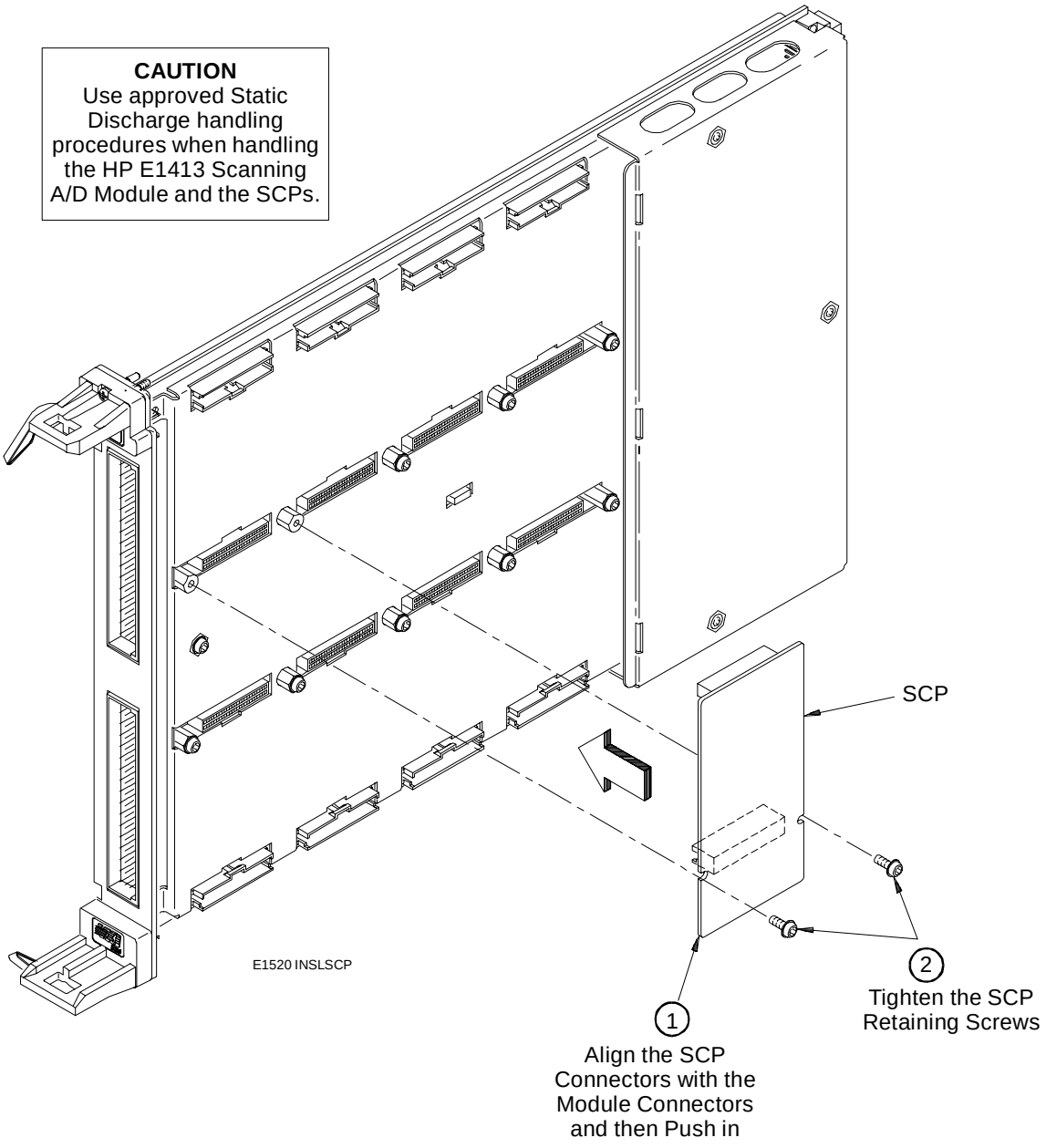
HP E1413 Step 1: Installing SCPs

Remove the cover from the HP E1413



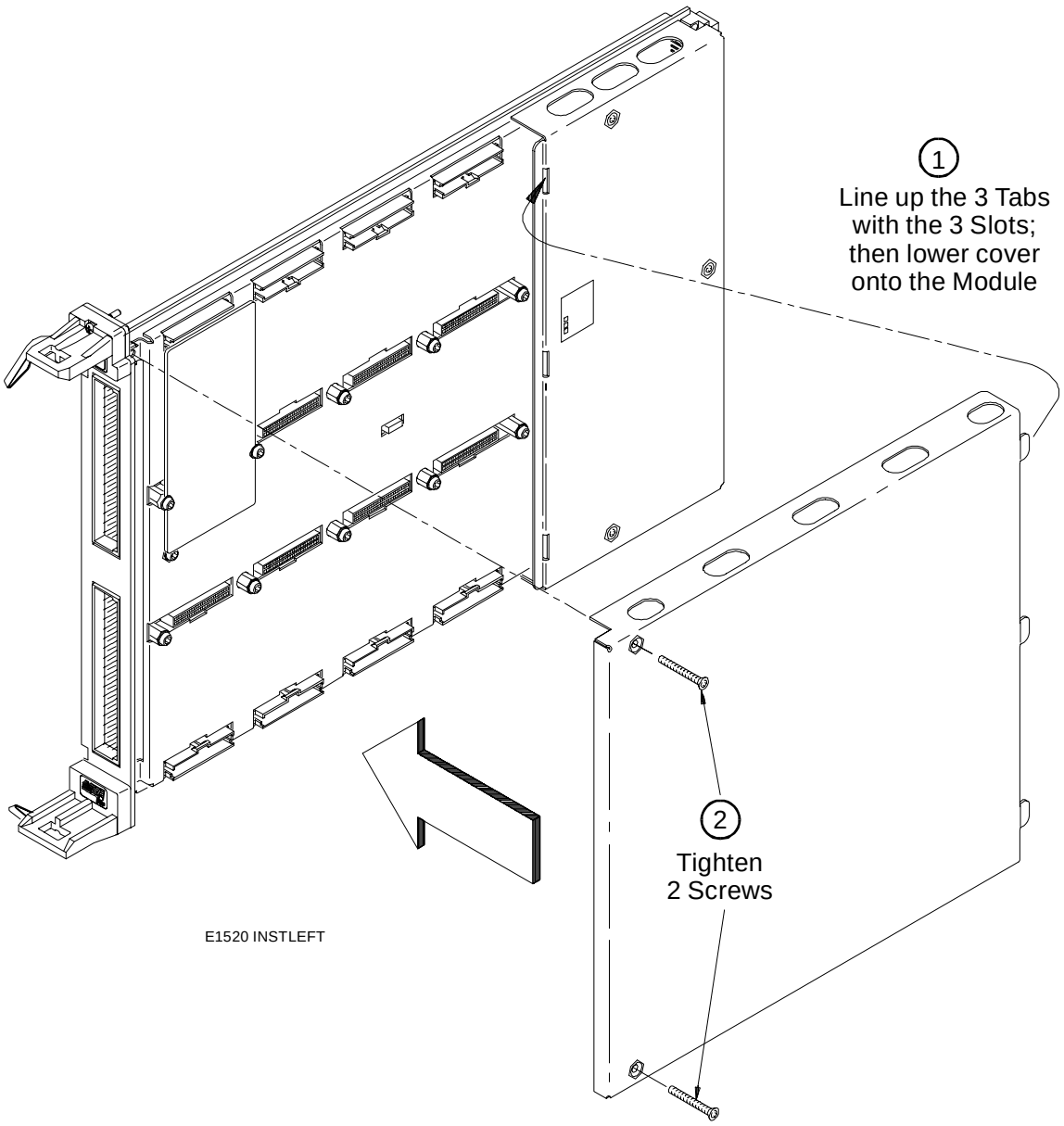
HP E1413 Step 2: Installing SCPs

CAUTION
Use approved Static Discharge handling procedures when handling the HP E1413 Scanning A/D Module and the SCPs.



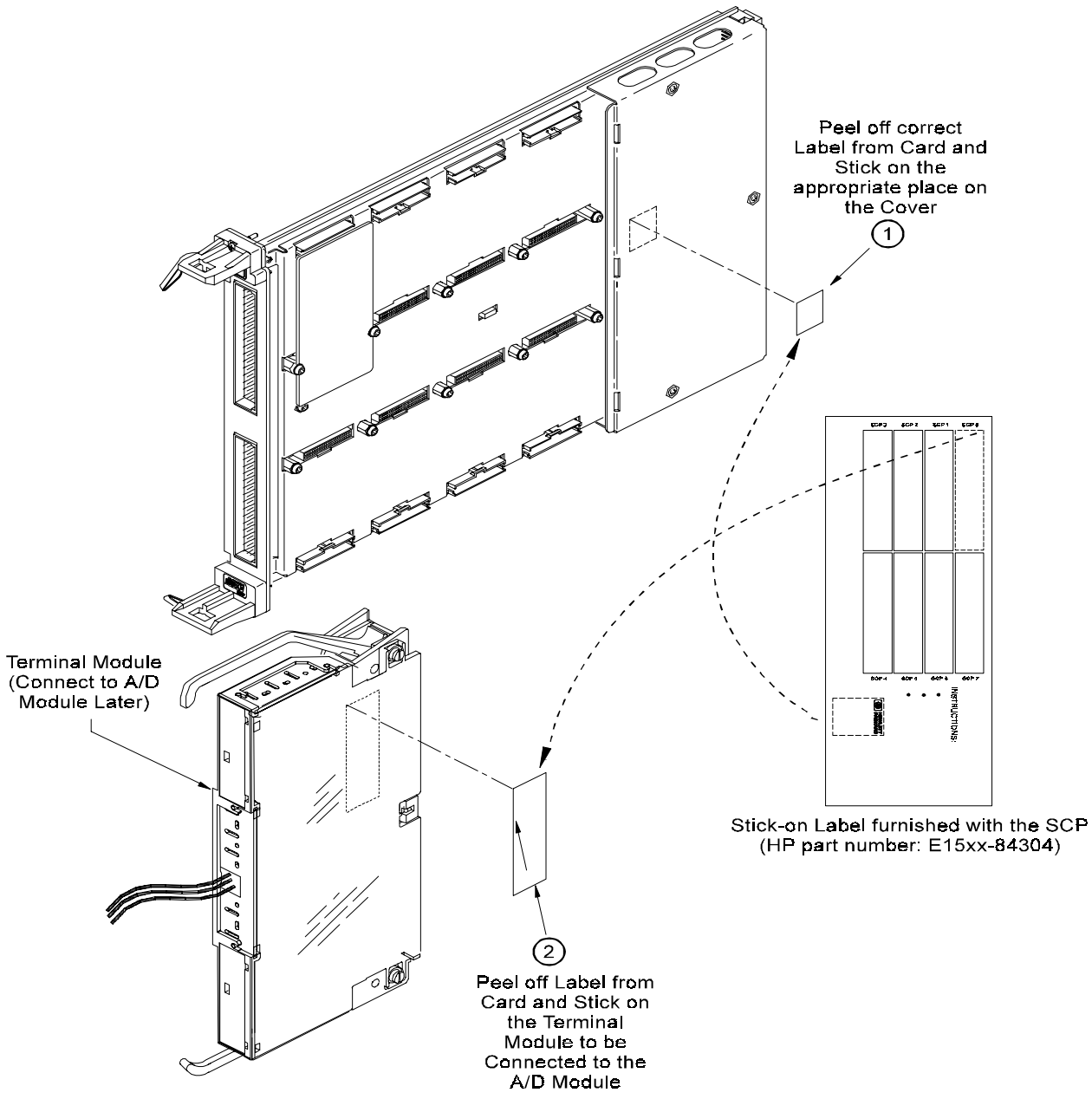
HP E1413 Step 3: Installing SCPs

Reinstall the cover on the HP E1413



HP E1413 Step 4: Installing SCPs

Apply label to the cover of the HP E1413



Disabling the Input Protect Feature (optional)

Disabling the input protect feature voids the HP E1313/E1413 warranty. The input protect feature allows the HP E1313/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, ranging amplifier, and A/D from destructive voltage levels. The level that trips the protection function has been set to provide a high probability of protection. The voltage level that is certain to cause damage is somewhat higher. **If, in your application, the importance of completing a measurement run outweighs the added risk of damage to your HP E1313/E1413, you may choose to disable the input protect feature.**

VOIDS WARRANTY Disabling the input protection feature voids the HP E1313/E1413 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 Figures 1-3 and 1-4 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 means you will not be able to execute:

- The SCPI calibration command CALibration: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.

The multimeter you use during the periodic calibration cycle should be considered your calibration transfer standard. Have your Calibration Organization control unauthorized access to its calibration constants. See the *HP E1313/E1413 Service Manual* for complete information on HP E1313/E1413 periodic calibration.

If you must 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 Figure 1-4 for the location of JM2201.

See Figure 1-3 for the location of the HP E1313s flash memory protect switch.

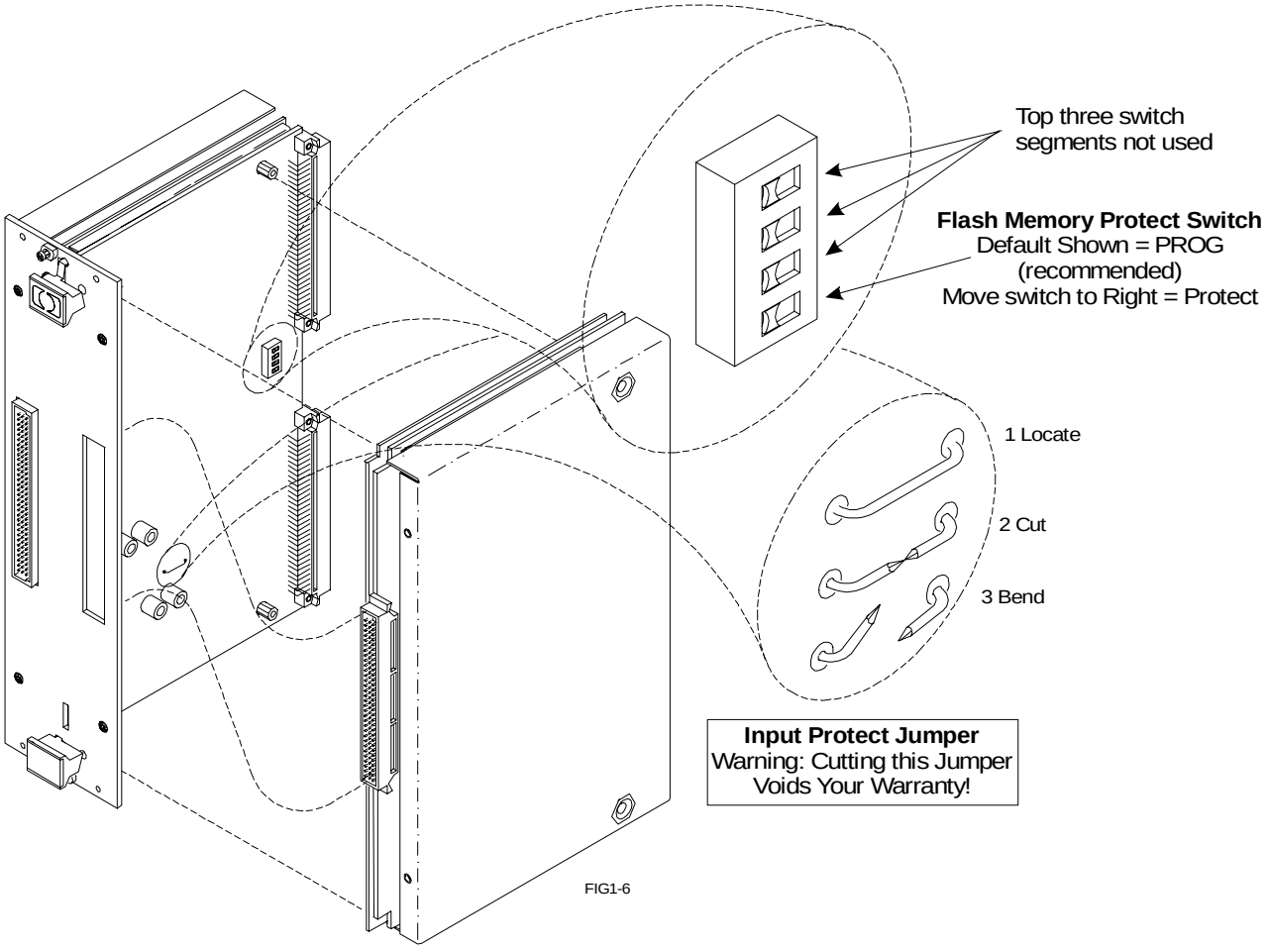


Figure 1-3. Locate and Access JM2201 and JM2202 on HP E1313

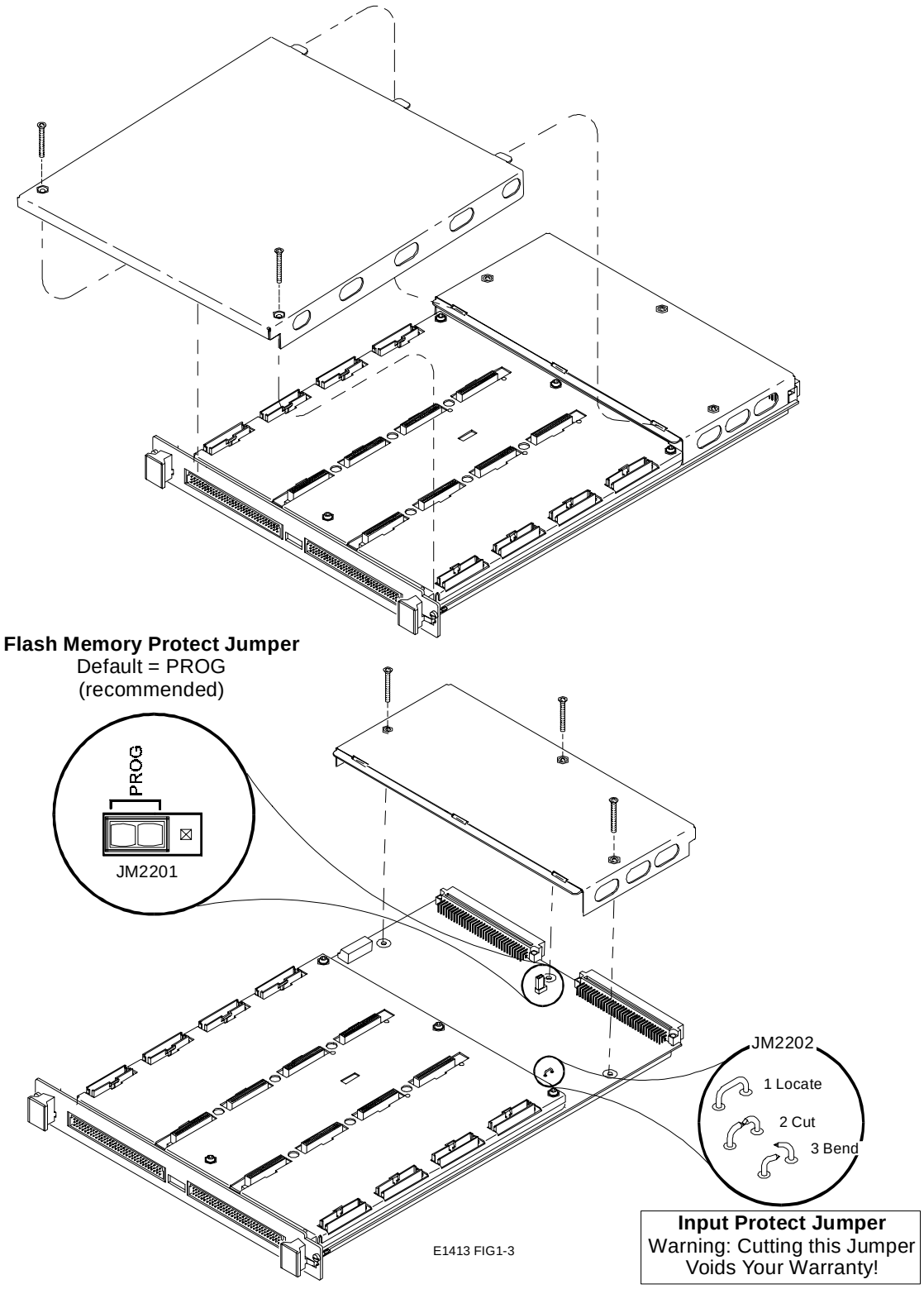


Figure 1-4. Locate and Access JM2201 and JM2202 on HP E1413

Instrument Drivers

If you will be using the HP E1313/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 E1300/E1301, HP E1306, and HP E1405B/E1406A, downloadable driver is supplied with your HP E1313/E1413. See the manual for your HP command module/mainframe for downloading procedures.

About Example Programs

Examples on Disc

All example programs mentioned by file name in this manual are available on the HP E1313/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 downloadable driver disc for the command module (HP E1405B or HP E1406A). An IBASIC version of the Verify program shown below is supplied on the LIF formatted downloadable command module driver.

Example Command Sequences

Where programming concepts are discussed in this manual, the commands to send to the HP E1313/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 E1313/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 105 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 E1313/E1413.

```
/* 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() /* Initialize 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;
}
```

Verifying a Successful Configuration

An example C-SCPI (Compiled-SCPI) program source is shown on the previous 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 Scanning A/D Converter module is operational and responding to commands. The program also has an error checking function (*check_error()*). It is important to include an instrument error checking routine in your programs, particularly your first trial programs so you get instant feedback while you are learning about the HP E1313/E1413. After you run the C-SCPI preprocessor and then compile and load this program, type *verif* to run the example.

Chapter 2
Field Wiring

About This Chapter

Except where noted, all references to the HP E1413 apply to the HP E1313. This chapter shows how to plan and connect field wiring to the HP E1313/E1413's terminal module. The chapter explains proper connection of analog signals to the HP E1413, both two-wire voltage type and four-wire resistance type measurements. Connections for other measurement types (for example, strain using the Bridge Completion SCPs) refer to the specific SCP manual in the "SCP Manuals" section. Chapter contents include:

Note An example C-SCPI program entitled *wiretest.cs* is included on the C-SCPI driver tape. After completing your field wiring, use this program to check for bad connections. The program performs "Open Transducer Detection" (see DIAGnostic:OTDetect[:STATe] 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 E1313/E1413 makes no assumptions about the relationship between Signal Conditioning Plug-on (SCP) function and the position in the HP E1313/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 Scanning A/D Converter modules. The following discussions will help you understand these factors.

SCP Positions and Channel Numbers

The HP E1313/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.

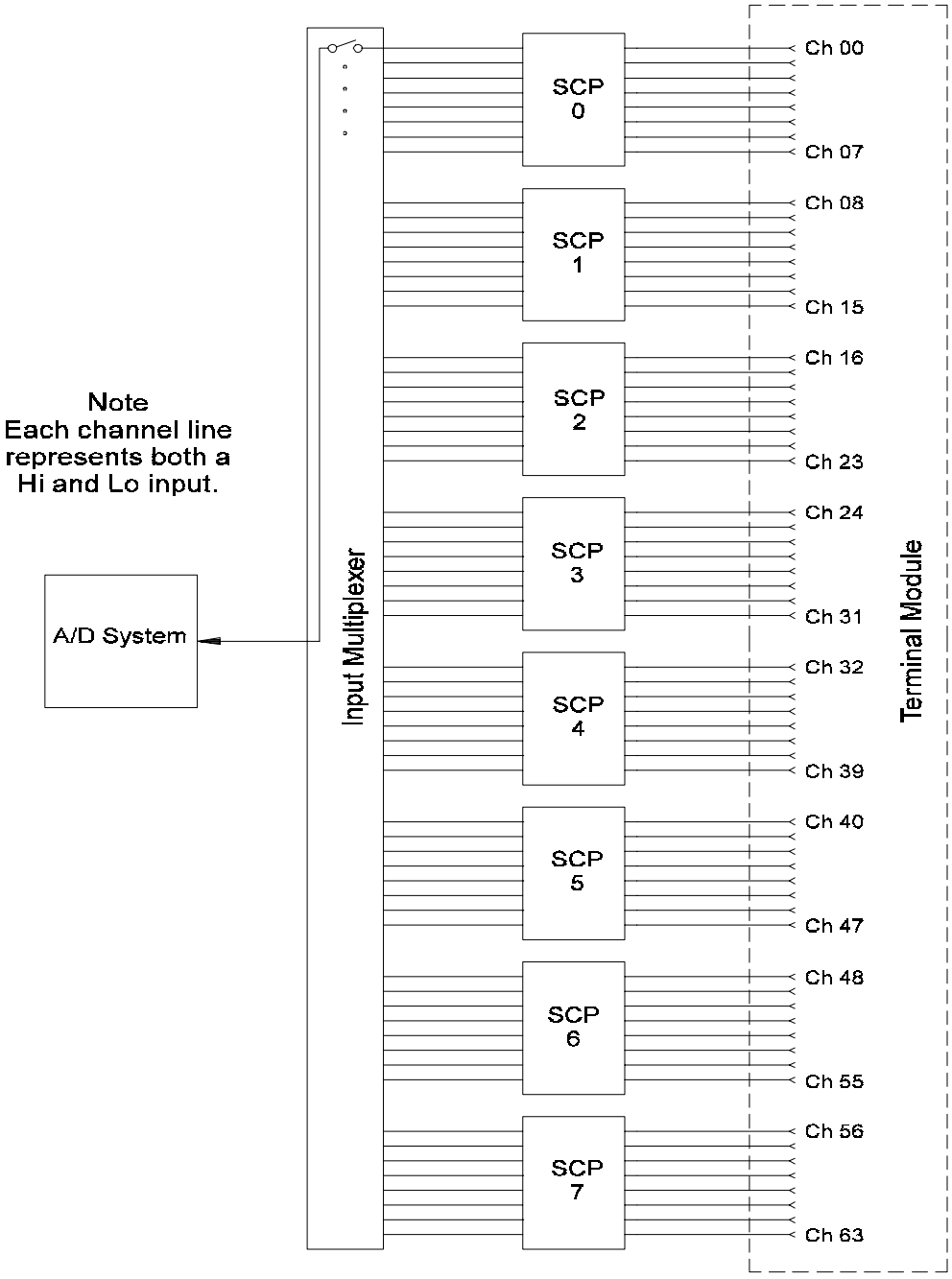


Figure 2-1. Channel Numbers at SCP Positions

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 E1313/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 E1313/E1413.

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

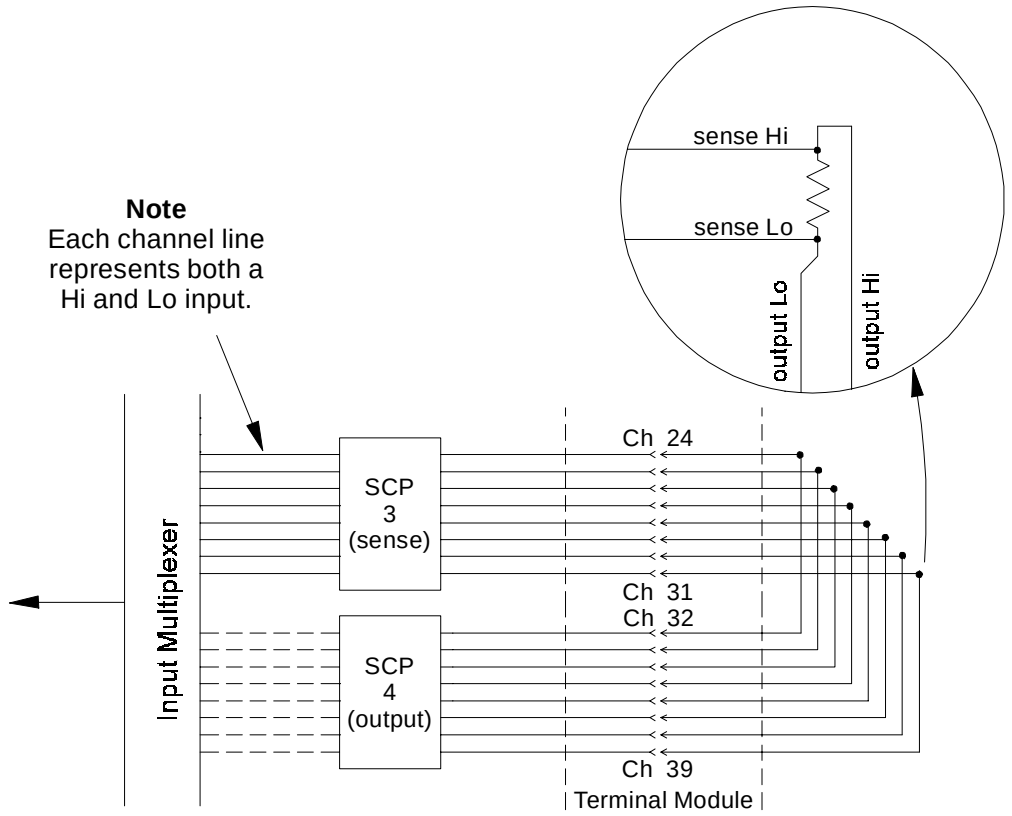


Figure 2-2. Pairing Output and Sense SCP Channels

Planning for Thermocouple Measurements

You can wire your thermocouples and your thermocouple reference temperature sensor to any of the HP E1313/E1413 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.

External wiring and connections to the HP E1313/E1413 are made using the terminal module (see pages 49 - 51).

Note

The isothermal reference temperature measurement made by an HP E1313/E1413 applies only to thermocouple measurements made by that instrument. In systems with multiple HP Scanning A/D Converter modules, each instrument must make its own reference measurements. The reference measurement made by one HP E1313/E1413 cannot be used to compensate thermocouple measurements made by another HP E1313/E1413.

IMPORTANT!

To make good low-noise measurements you must use shielded wiring from the device under test to the terminal module at the HP E1313/E1413. The shield must be continuous through any wiring panels or isothermal reference connector blocks and must be grounded at a single point to prevent ground loops. See "Preferred Measurement Connections" on page 43 for more information.

Terminal Modules

The HP E1313/E1413 is comprised of an A/D module and a spring-clamp type terminal module. The screwless terminals utilize a spring-clamp terminal for connecting solid or stranded wire. Connection is made with a simple push of a three-pronged insertion tool (HP part number 8710-2127), which is shipped with the HP E1313/E1413. If the spring-clamp type terminal module is not desired, a crimp-and-insert terminal module (Option A3E) and an interface to rack mount terminal panel (Option A3F) is available for the HP E1313 and HP E1413 (see page 56).

The terminal module provides the following:

- Terminal connections to field wiring.
- Strain relief for the wiring bundle.
- Reference junction temperature sensing for thermocouple measurements.
- Ground to Guard connections for each channel.

The SCPs and Terminal Module

The same terminal is used for all field wiring regardless of which Signal Conditioning Plug-on (SCP) is used. Each SCP includes a set of labels to map that SCP's channels to the terminal module's terminal blocks. See Step 4 on page 21 for the HP E1313 terminal module and page 25 for the HP E1413 terminal module.

Terminal Module Layout

Figure 2-3 shows the HP E1313 terminal module and Figure 2-4 shows the HP E1413 terminal module.

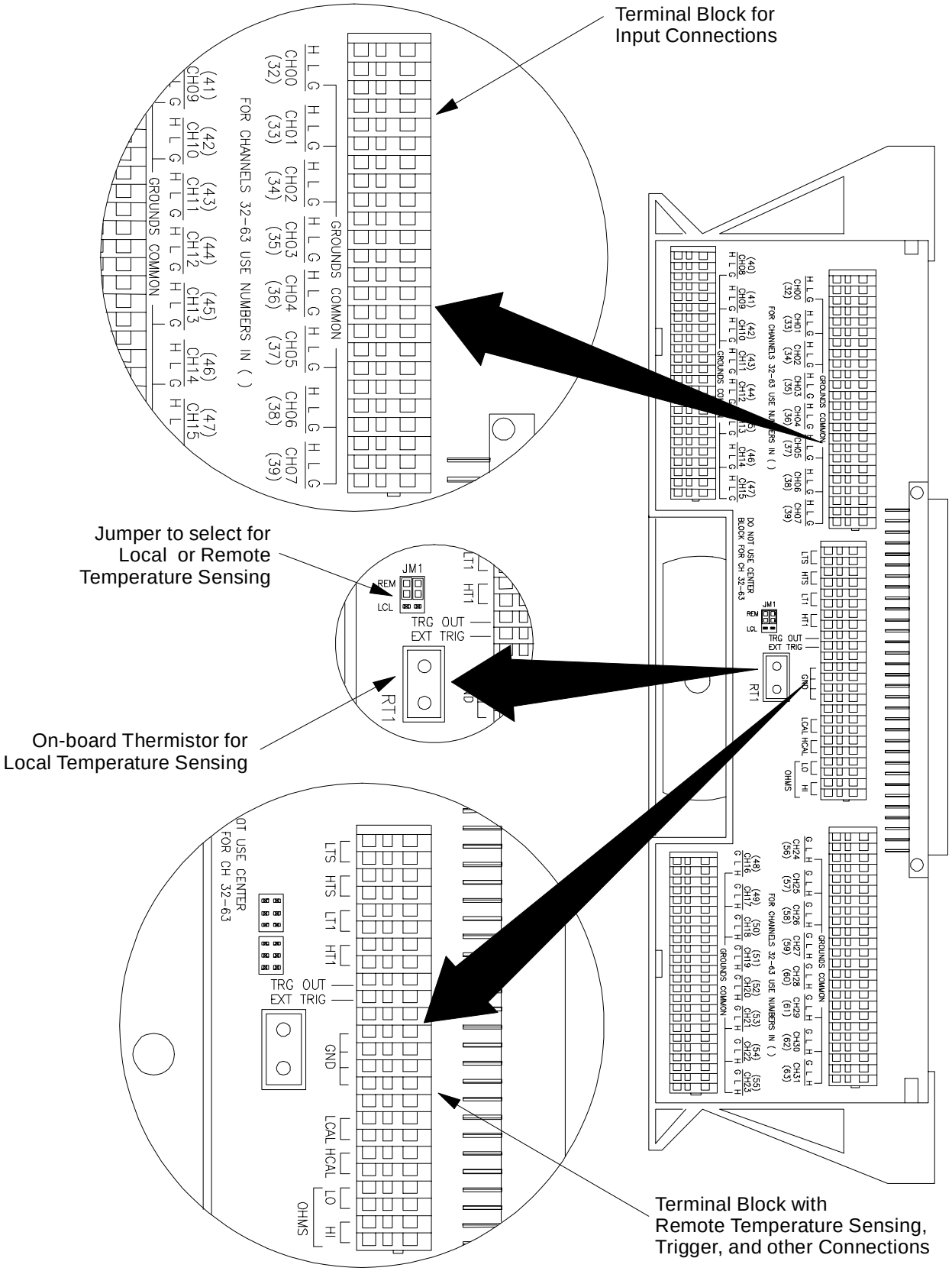


Figure 2-3. HP E1313 Terminal Module Layout

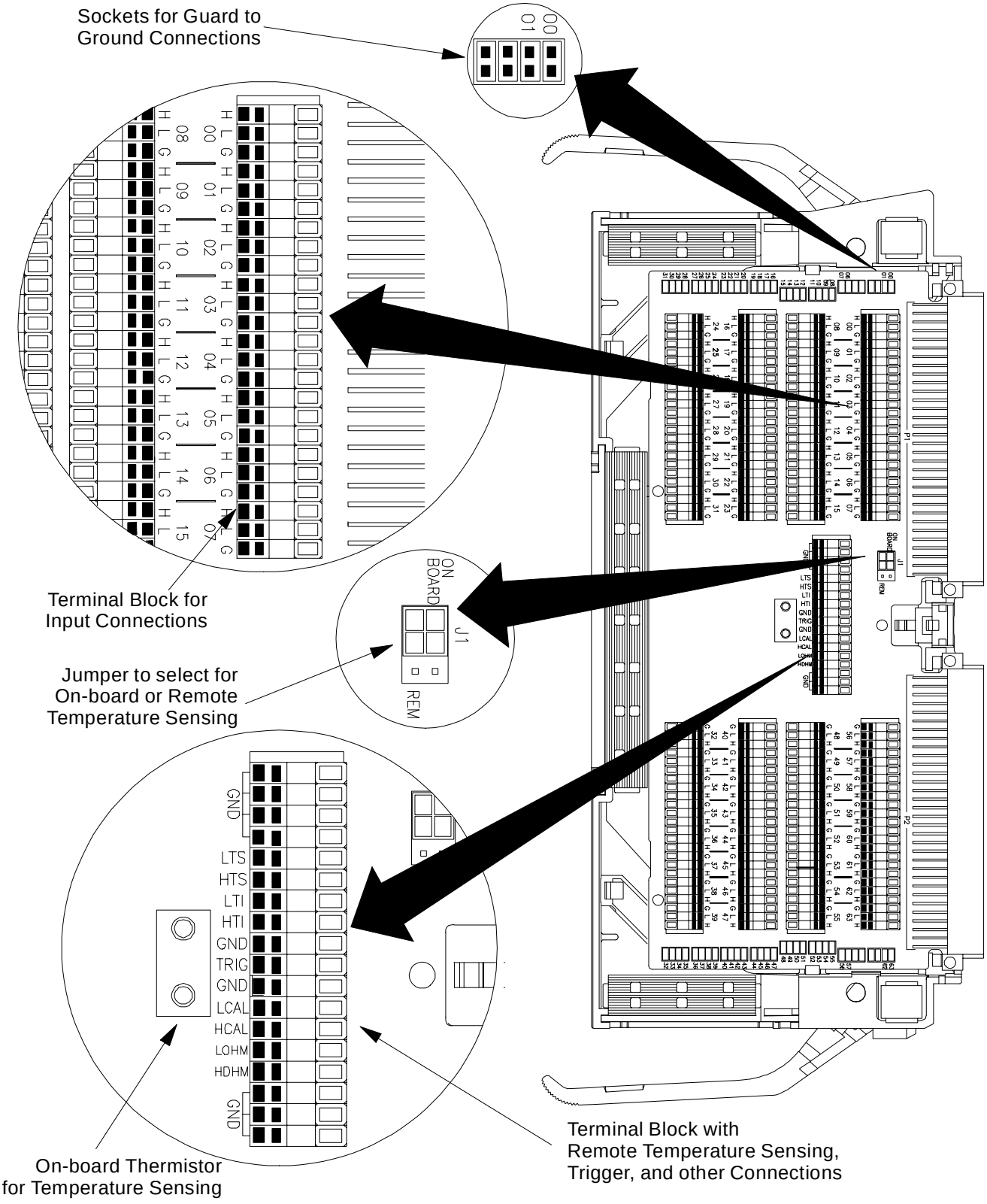


Figure 2-4. HP E1413 Terminal Module Layout

Reference Temperature Sensing with the HP E1313

The HP E1313 can have two terminal modules installed. To obtain the specified accuracy through each terminal module, the temperature of both must be sensed. A sense channel is required on each terminal module which leaves 31 channels on each terminal module for thermocouple and other measurements. Figure 2-5 shows how the HP E1313 internally connects its single current source to the two reference thermistors in series. Note that a diode provides a current path when terminal module 2 is absent, but that terminal module 1 must be in place and configured properly to provide a current path to terminal module 2. Figure 2-7 shows sensing the on-board thermistor. Refer to Figure 2-6 for remote thermistor connection.

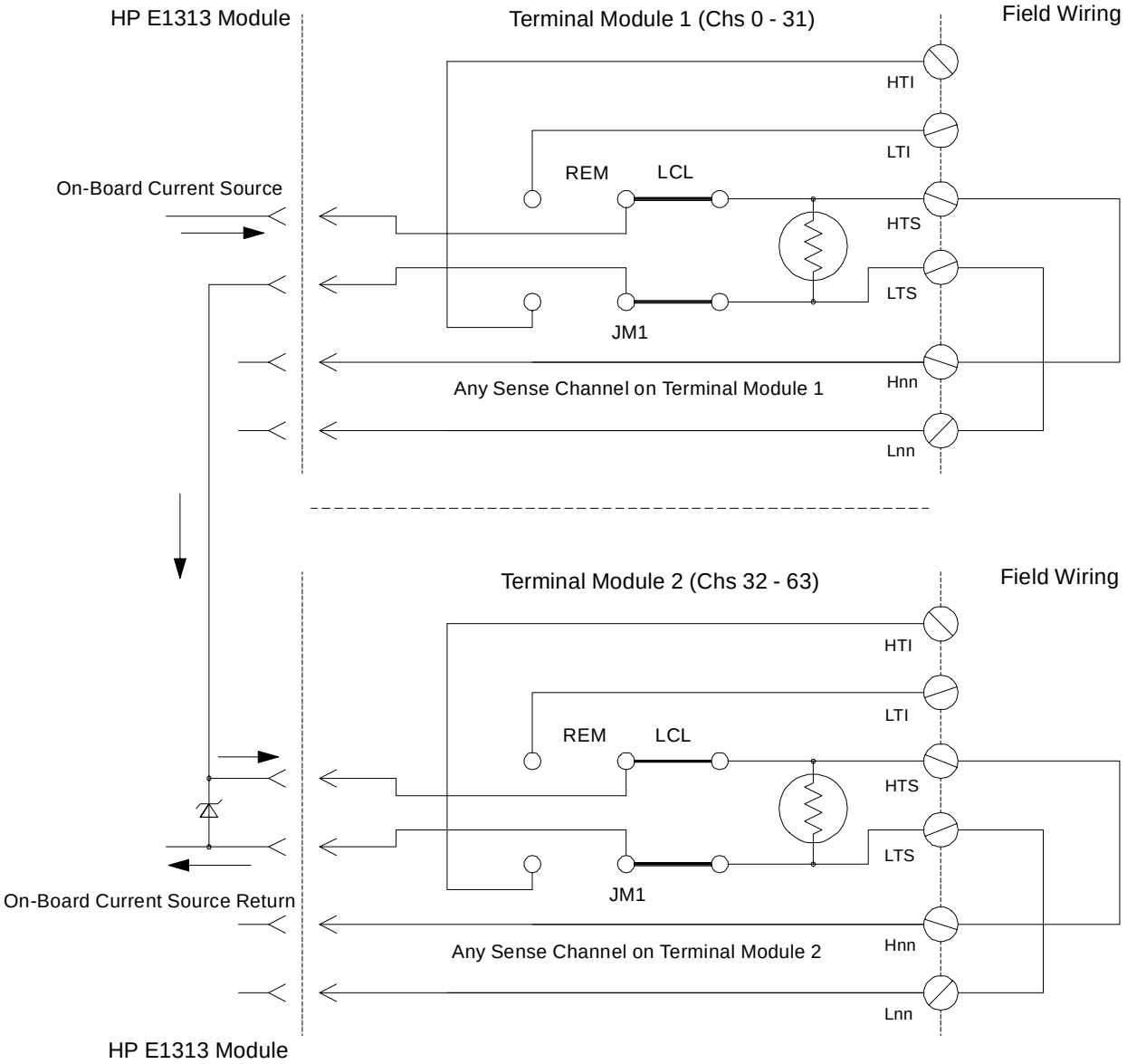


Figure 2-5. HP E1313 Reference Temperature Circuit

Reference Temperature Sensing with the HP E1413

The terminal module provides an on-board thermistor for sensing isothermal reference temperature of the terminal blocks. Also provided is a jumper set (J1 in Figure 2-4) to route the HP E1413's on-board current source to a thermistor or RTD on a remote isothermal reference block. Figures 2-6 and 2-7 show connections for both local and remote sensing.

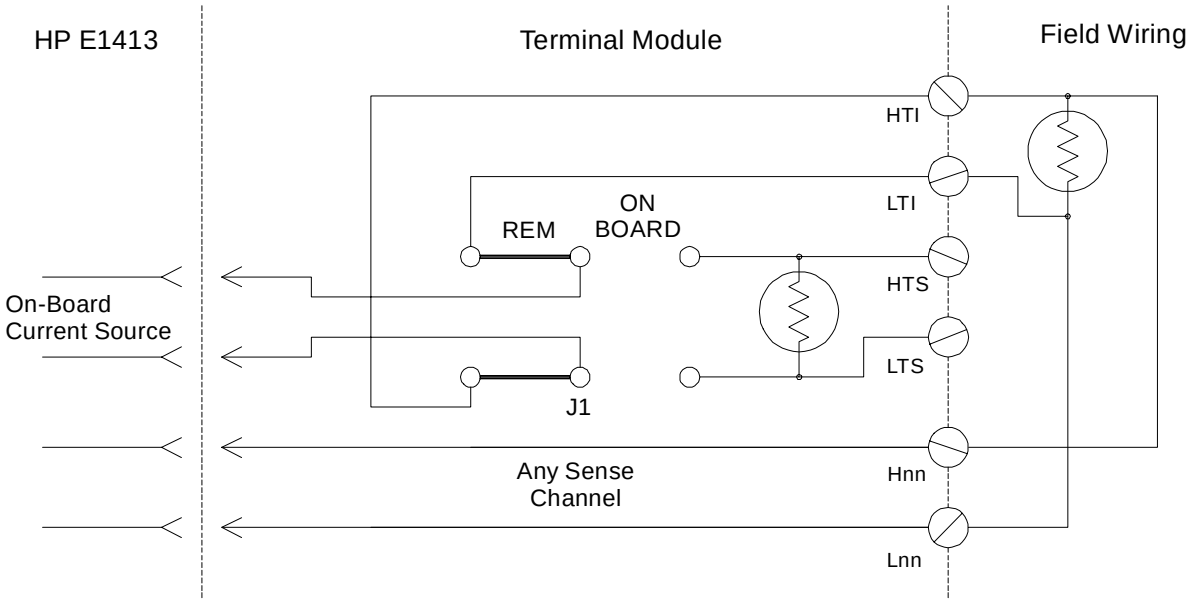


Figure 2-6. Remote Thermistor or RTD Connections

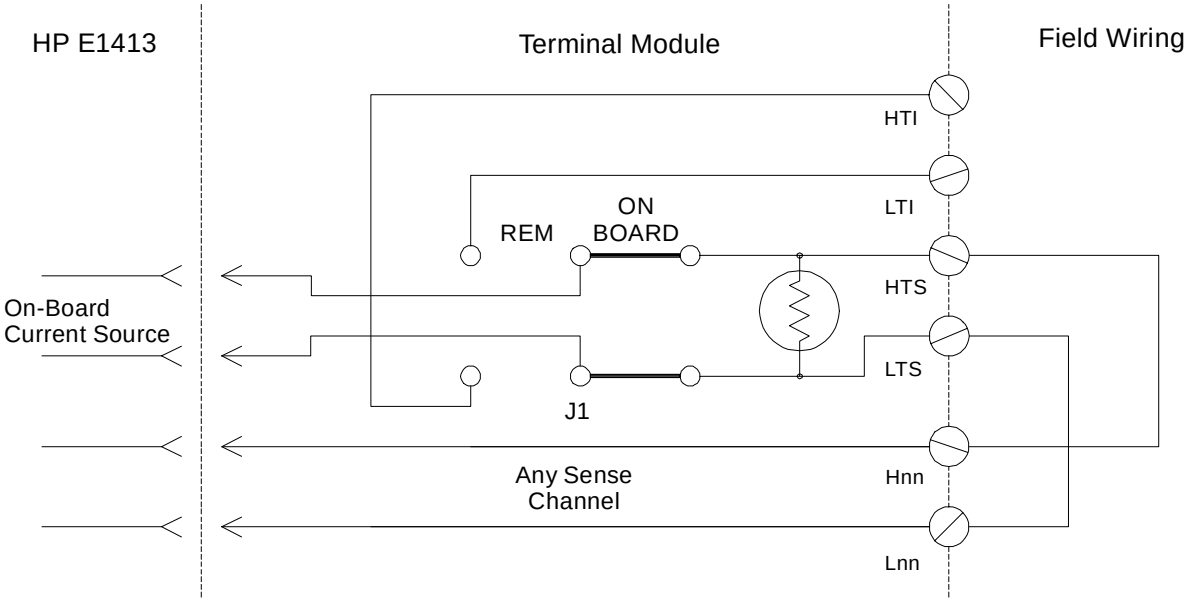


Figure 2-7. On-Board Thermistor Connections

Terminal Module Considerations for TC Measurements

The isothermal characteristics of the HP E1313/E1413 terminal module are crucial for good TC readings and can be affected by any of the following factors:

1. The clear plastic cover must be on the terminal module.
2. The thin white mylar thermal barrier must be inserted over the terminal module connector (HP E1413 only). This prevents airflow from the HP E1413 module into the terminal module.
3. The terminal module must also be in a fairly stable temperature environment, and it is best to minimize the temperature gradient between the HP E1413 module and the terminal module.
4. The VXI mainframe cooling fan filters must be clean and there should be as much clear space in front of the fan intakes as possible.
5. Recirculating warm air inside a closed-rack cabinet can cause a problem if the terminal module is suspended into ambient air that is significantly warmer or cooler. If the mainframe recess is mounted in a rack with both front and rear doors, closing both doors helps keep the entire HP E1413 at a uniform temperature. If there is no front door, try opening the back door.
6. HP recommends that the cooling fan switch on the back of the HP E1401 Mainframe is in the "High" position. The normal variable speed cooling fan control can make the internal HP E1413 module temperature cycle up and down, which affects the amplifiers with these µV level signals.

Scanning Two Reference Temperature Channels

The following simple command sequence shows linking channels to EU conversions to measure the reference temperature of both of the HP E1313's terminal modules and to measure thermocouples on the remaining channels.

Link reference temperature to channels 0 and 32 (terminal module 1 and 2).

```
SENS:REF  THER,5000,AUTO,(@100,132)  SENS:FUNC:TEMP
```

Link E type thermocouple measurements to channels 1 - 31 and 33 - 63.

```
SENS:FUNC:TEMP  TC,E,AUTO,(@101:131,133:163)
```

Define scan list 1 to measure the reference temperature of terminal module 1. Then measure thermocouples on channels 1 - 31, measure the ref temp of terminal module 2, then measure thermocouples on channels 33 - 63.

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

Now select LIST1 as the current scan list.

```
ROUT:SCAN  LIST1
```

When the module is triggered, it will scan channels 0 through 63. Since the first channel on each terminal module measures the reference temperature of that module, the thermocouple measurements that follow will be referenced against the temperature of their respective terminal modules.

Preferred Measurement Connections

IMPORTANT!

For any Scanning A/D Converter module to scan channels at high speeds, it must use a very short sample period (<10 μsecond for the HP E1413). If significant normal-mode noise is presented to its inputs, that noise will be part of the measurement. To make quiet, accurate measurements in electrically noisy environments, use properly connected shielded wiring between the A/D and the device under test. Figure 2-8 shows recommended connections for powered transducers, thermocouples, and resistance transducers. (See Appendix F for more information on wiring techniques).

Hints

1. Use individually shielded, twisted-pair wiring for each channel.
2. Connect the shield of each wiring pair to the corresponding Guard (G) terminal on the terminal module (see Figure 2-9 for schematic of Guard to Ground circuitry on the terminal module).
3. The terminal module is shipped with the Ground-Guard (GND-GRD) shorting jumper installed for each channel. These may be left installed or removed (see Figure 2-10 to remove the jumper), dependent on the following conditions:
   a. **Grounded Transducer with shield connected to ground at the transducer:** Low frequency ground loops (DC and/or 50/60 Hz) can result if the shield is also grounded at the terminal module end. To prevent this, remove the GND-GRD jumper for that channel (Figure 2-8 A/C).
   b. **Floating Transducer with shield connected to the transducer at the source:** In this case, the best performance will most likely be achieved by leaving the GND-GRD jumper in place (Figure 2-8 B/D).
4. In general, the GND-GRD jumper can be left in place unless it is necessary to remove to break low frequency (below 1 kHz) ground loops.
5. Use good quality foil or braided shield signal cable.
6. Route signal leads as far as possible from the sources of greatest noise.
7. In general, do not connect Hi or Lo to Guard or Ground at the HP E1413.
8. It is best if there is a DC path somewhere in the system from Hi or Lo to Guard/Ground.
9. The impedance from Hi to Guard/Ground should be the same as from Lo to Guard/Ground (balanced).
10. Since each system is different, do not be afraid to experiment using the suggestions presented here until you find an acceptable noise level.

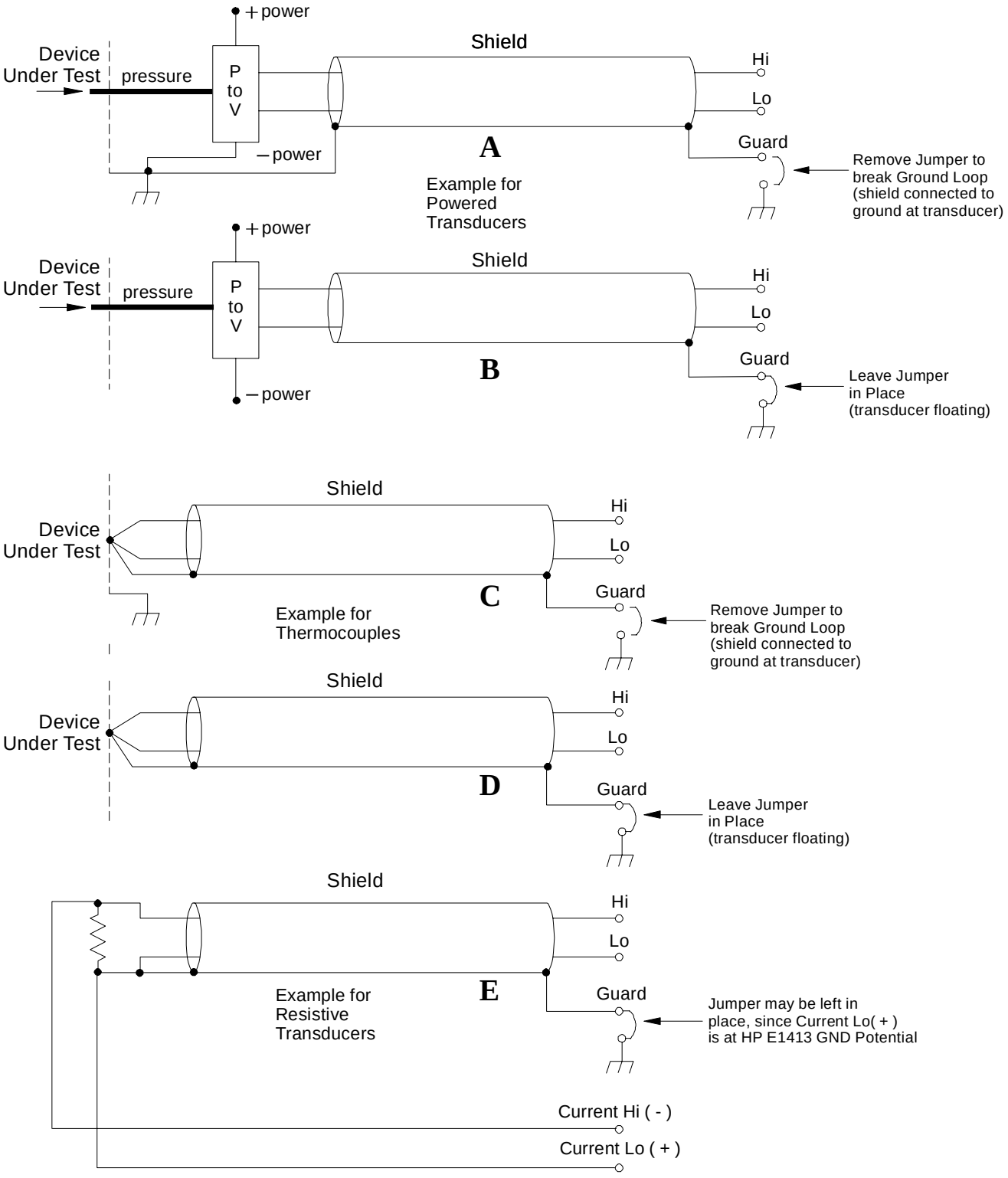


Figure 2-8. Preferred Signal Connections

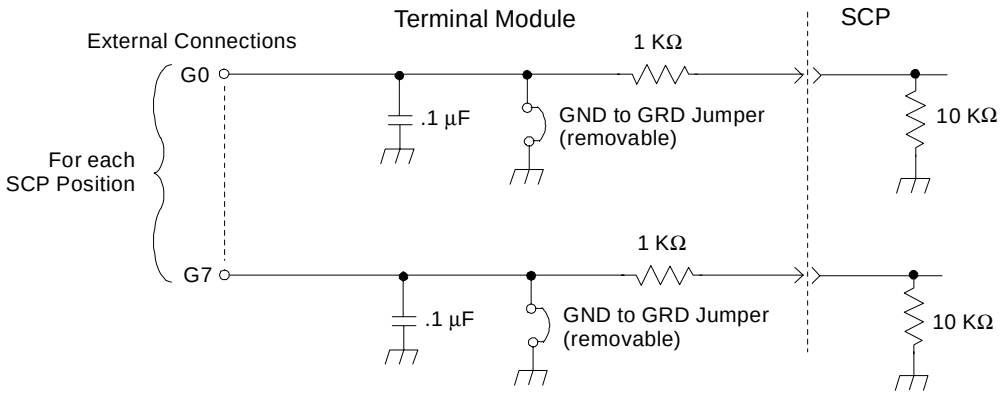


Figure 2-9. GRD/GND Circuitry on Terminal Module

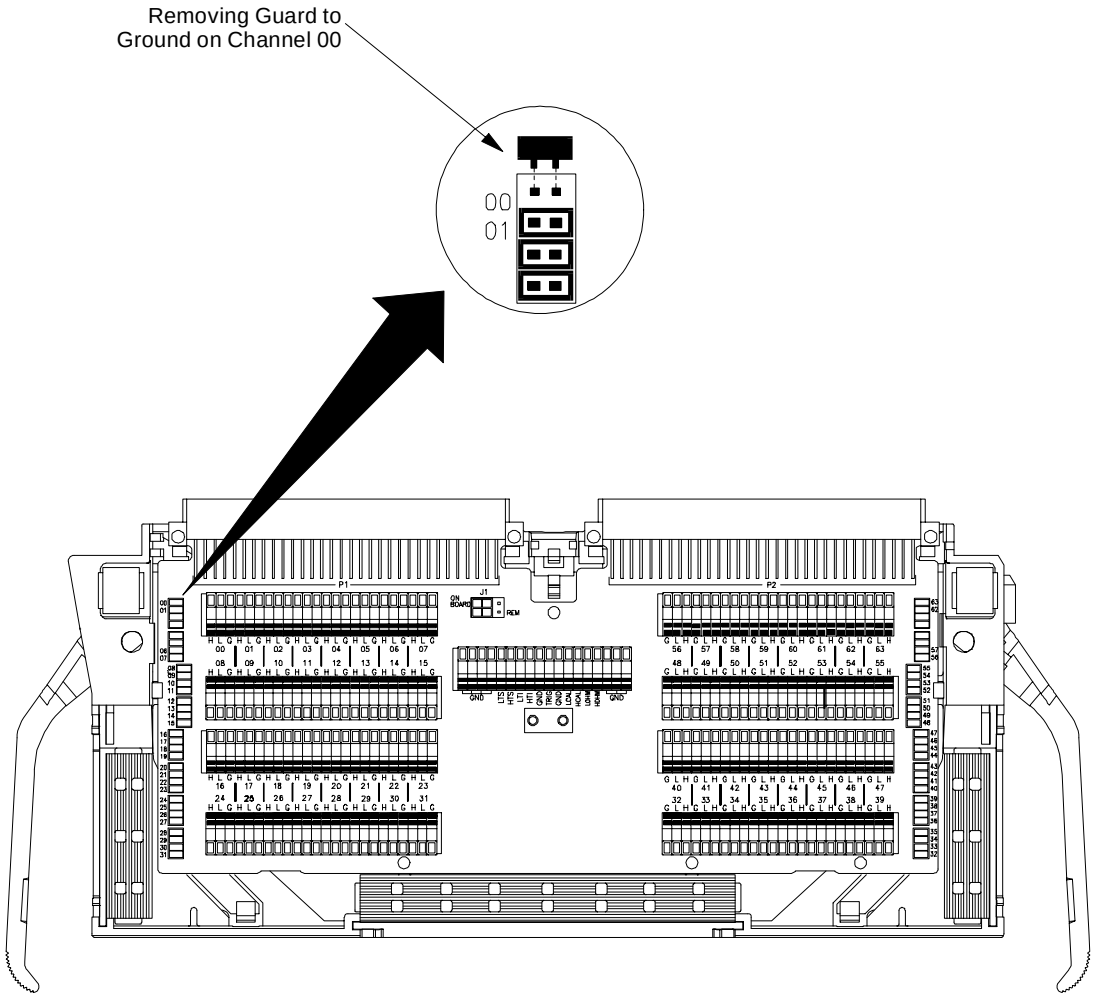


Figure 2-10. Grounding the Guard Terminal

Connecting the On-Board Thermistor

Figures 2-11 and 2-12 show how to use the HP Scanning A/D Converter module to make temperature measurements with or without using the on-board thermistor. The thermistor is used for reference junction temperature sensing for thermocouple measurements. Figure 2-11 shows the configuration for the HP E1313 terminal module and Figure 2-12 shows the configuration for the HP E1413 terminal module.

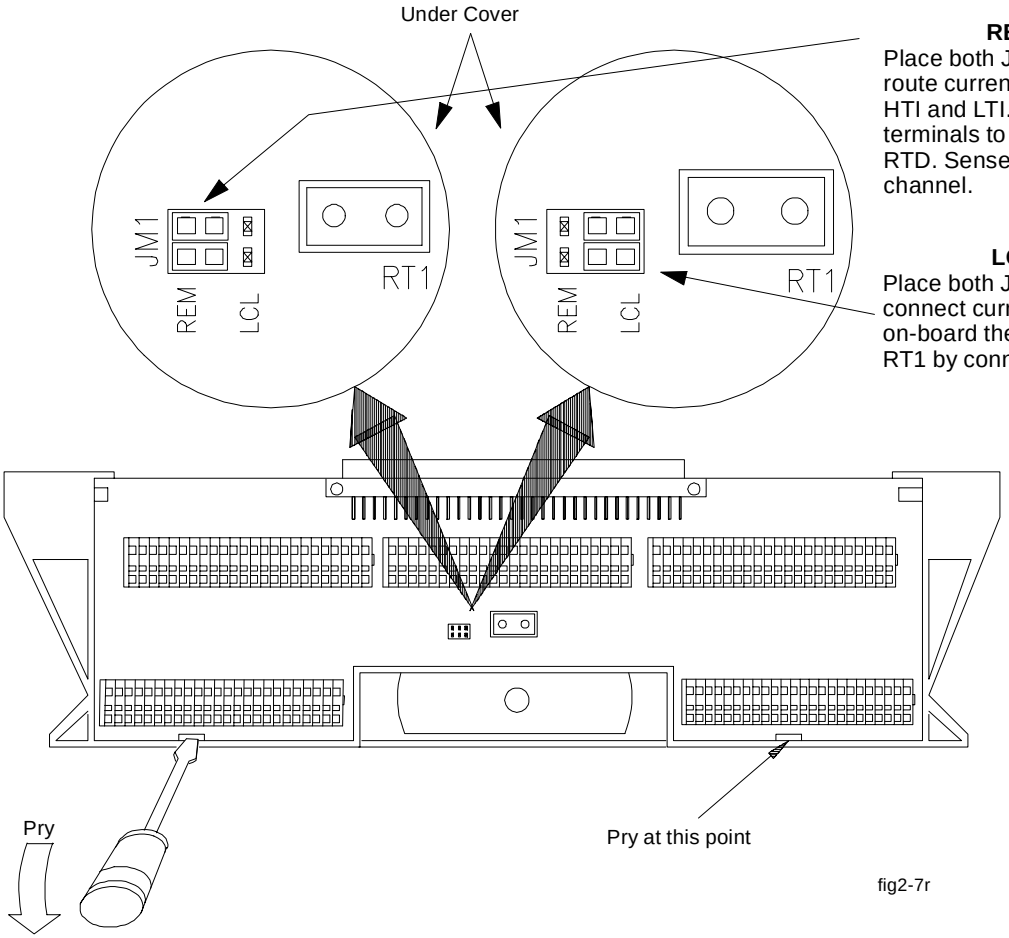
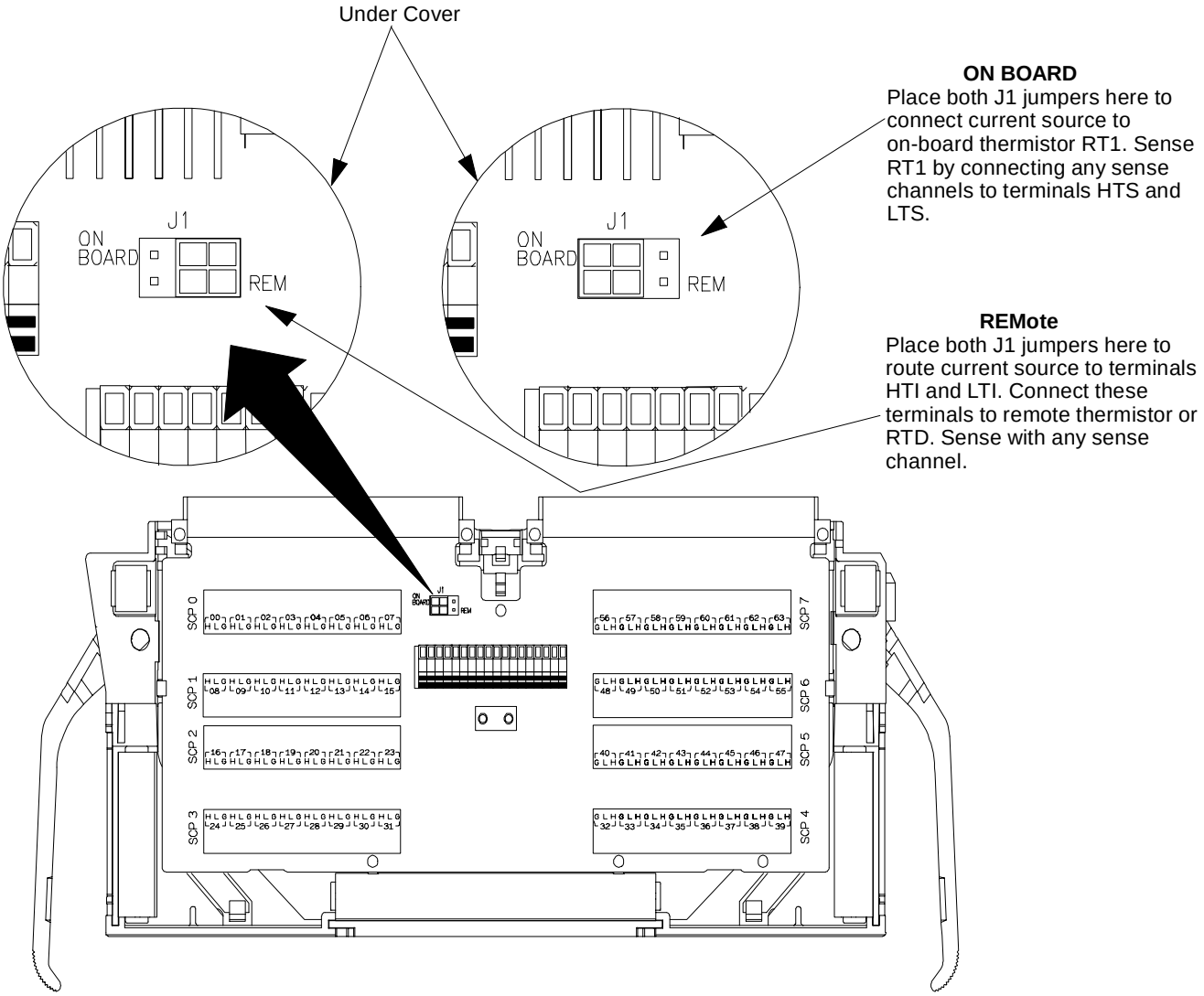


Figure 2-11. Temperature Sensing for HP E1313 Terminal Module



See page 50 to remove the cover

Figure 2-12. Temperature Sensing for HP E1413 Terminal Module

Adding Components to the HP E1413 Terminal Module

The back of the terminal module PC board provides surface mount pads which you can use to add serial and parallel components to any channel's signal path. Figure 2-13 shows additional component locator information (see the schematic and pad layout information on the back of the teminal module PC board). Figure 2-14 shows some usage example schematics.

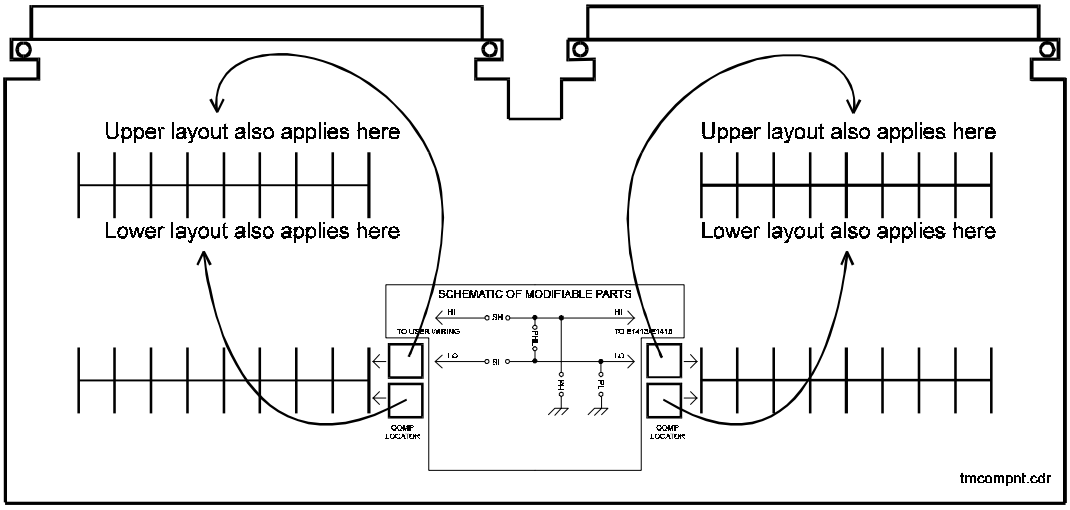


Figure 2-13. Additional Component Location Information

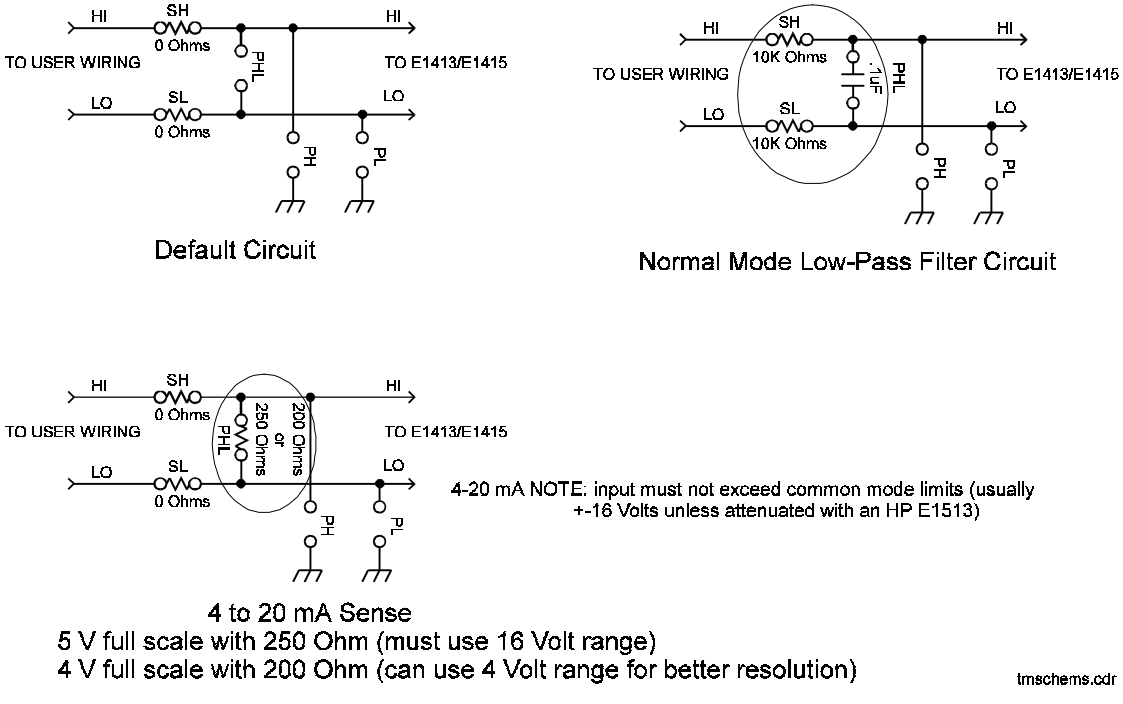


Figure 2-14. Serial and Parallel Component Examples

Wiring/Attaching the HP E1313 Terminal Module

Figure 2-15 shows how to wire and attach an HP E1313 terminal module.

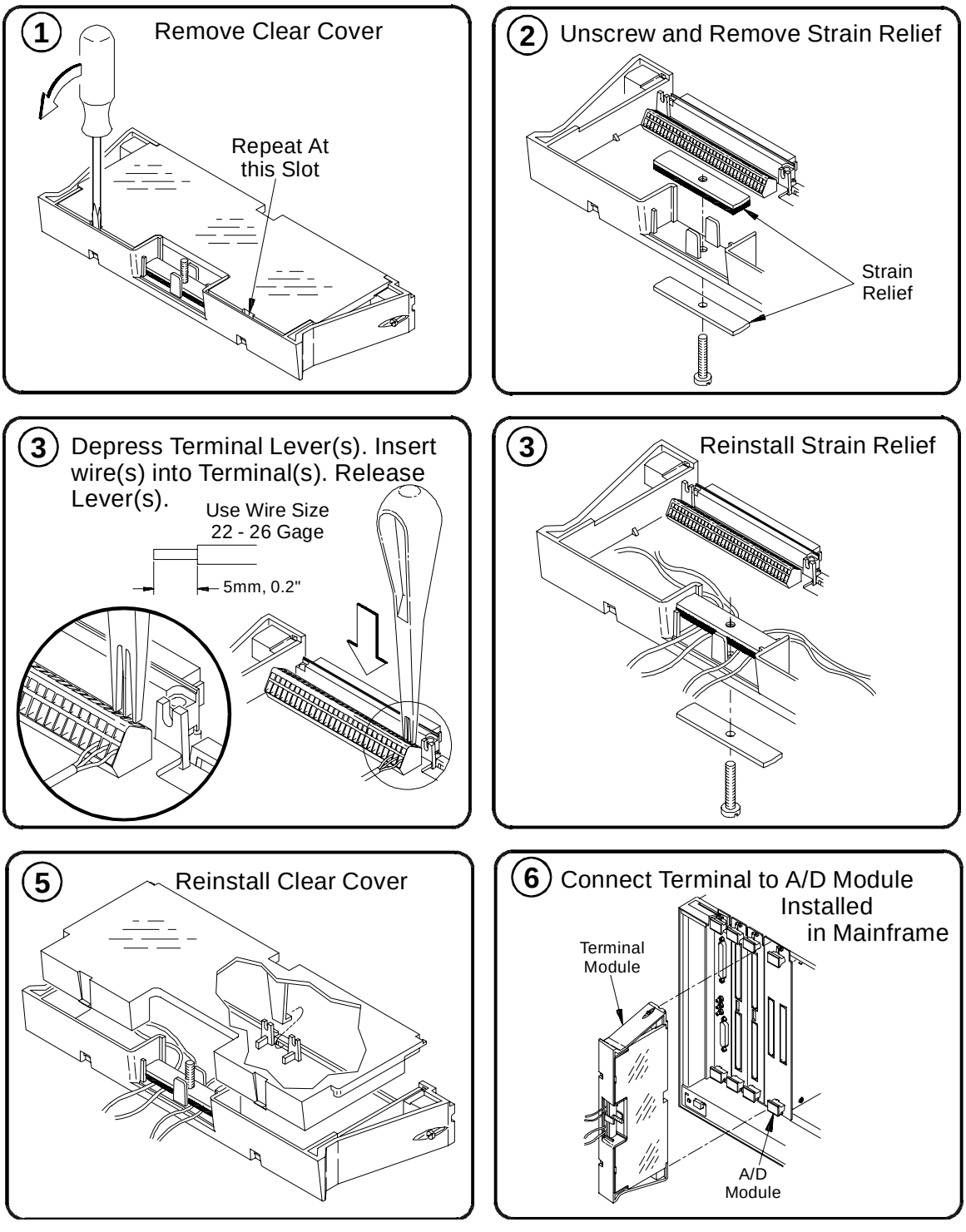


Figure 2-15. Wiring and Connecting the HP E1313 Terminal Module

Wiring the HP E1413 Terminal Module

Figure 2-16 shows how to open and wire the HP E1413 terminal module.

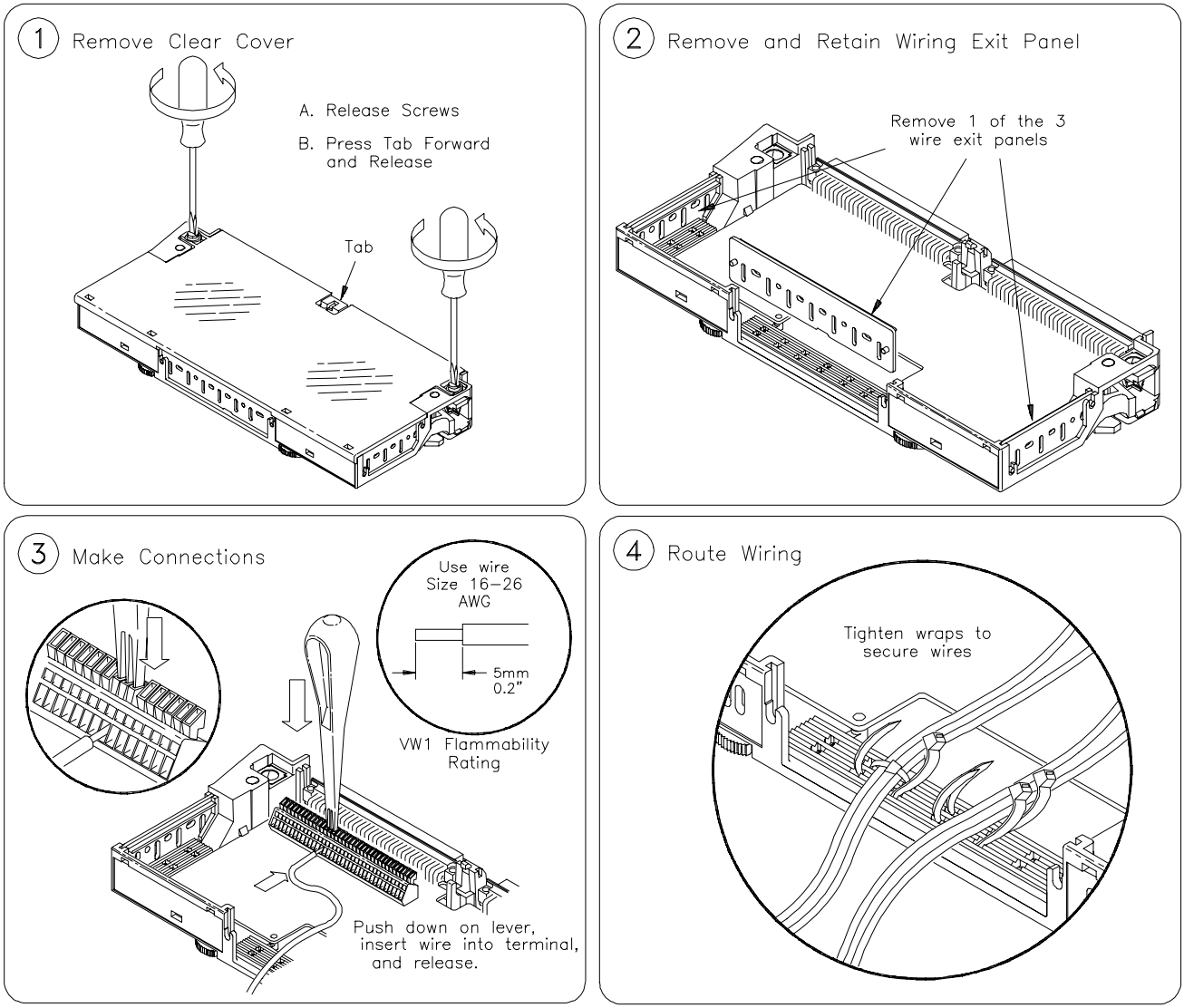


Figure 2-16. Wiring the HP E1413 Terminal Module

Continued on Next Page

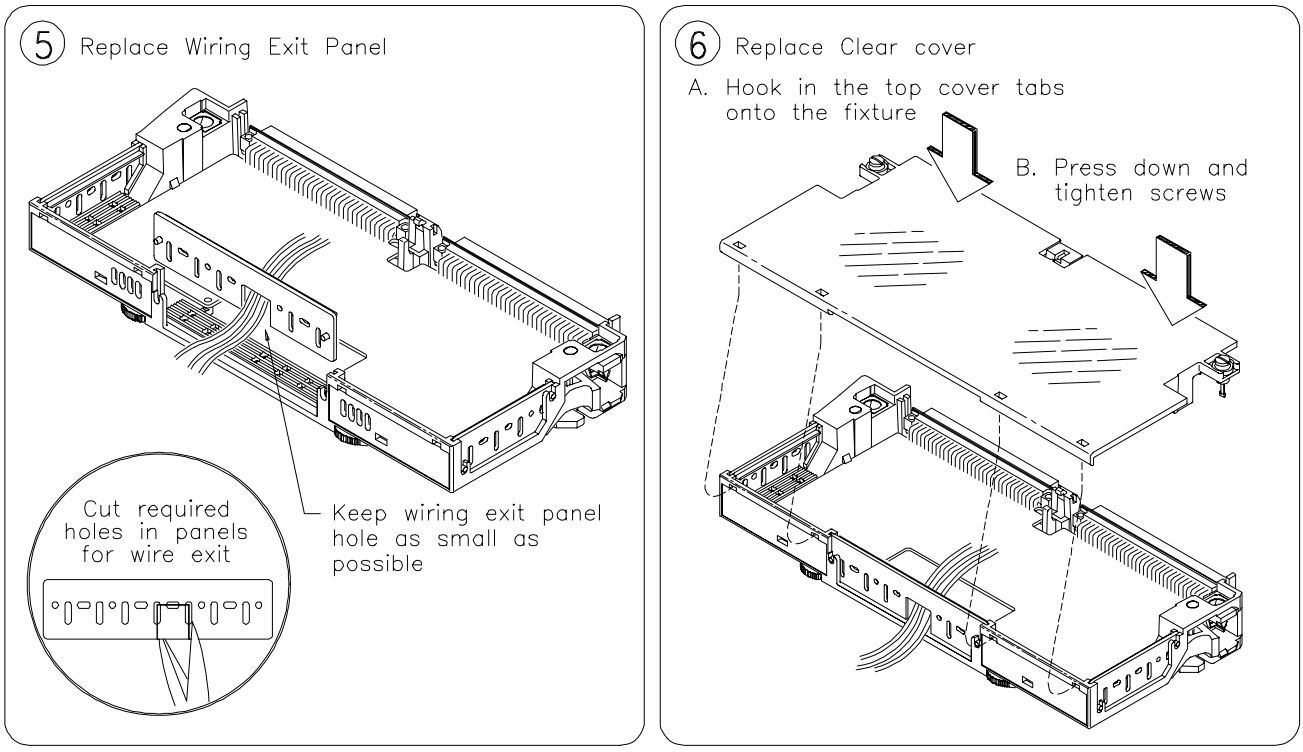


Figure 2-16 (continued). Wiring the HP E1413 Terminal Module

Attaching the HP E1413 Terminal Module

Figure 2-17 shows how to attach the HP E1413 terminal module.

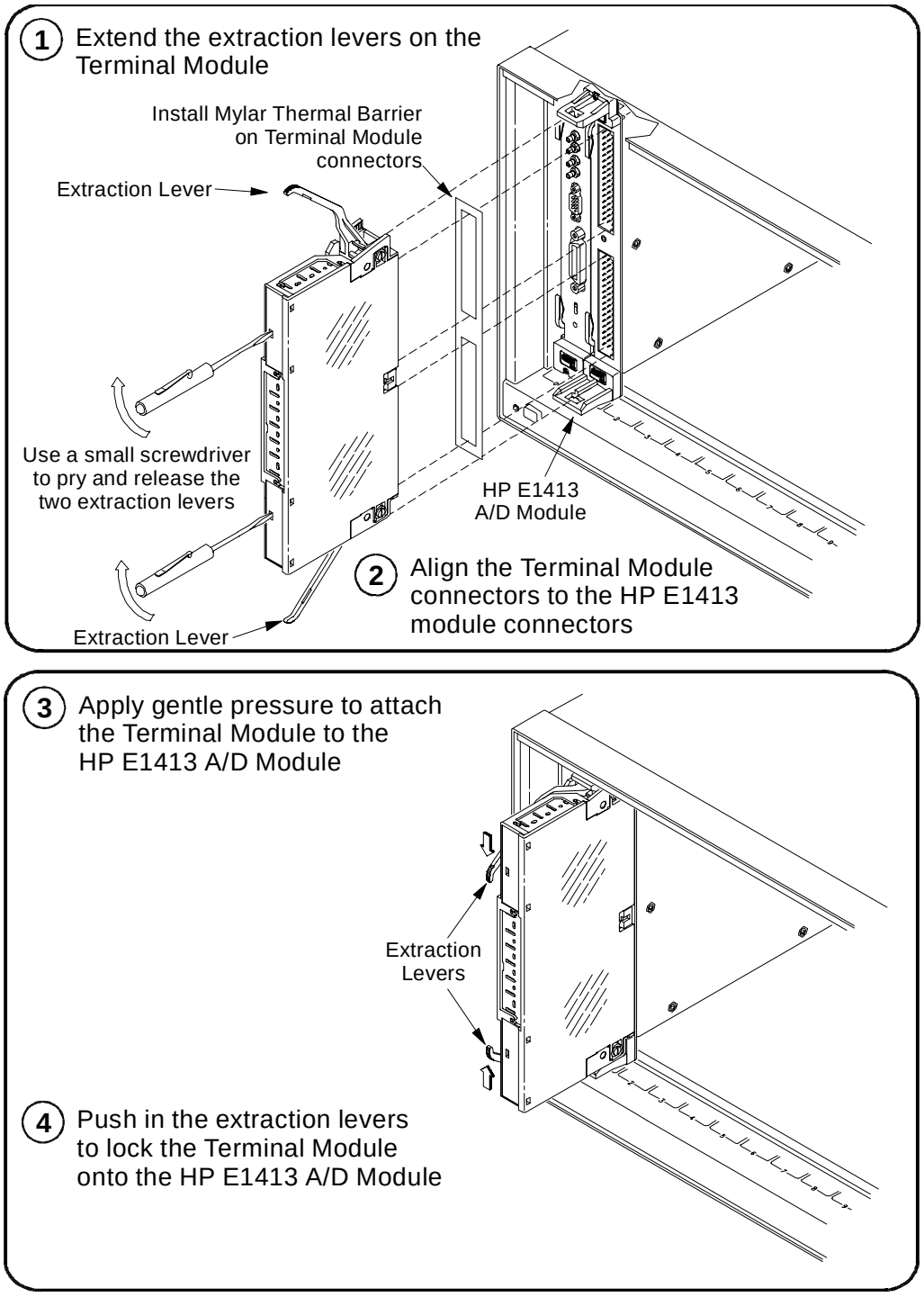


Figure 2-17. Attaching the HP E1413 Terminal Module

Removing the HP E1413 Terminal Module

Figure 2-18 shows how to remove the HP E1413 terminal module.

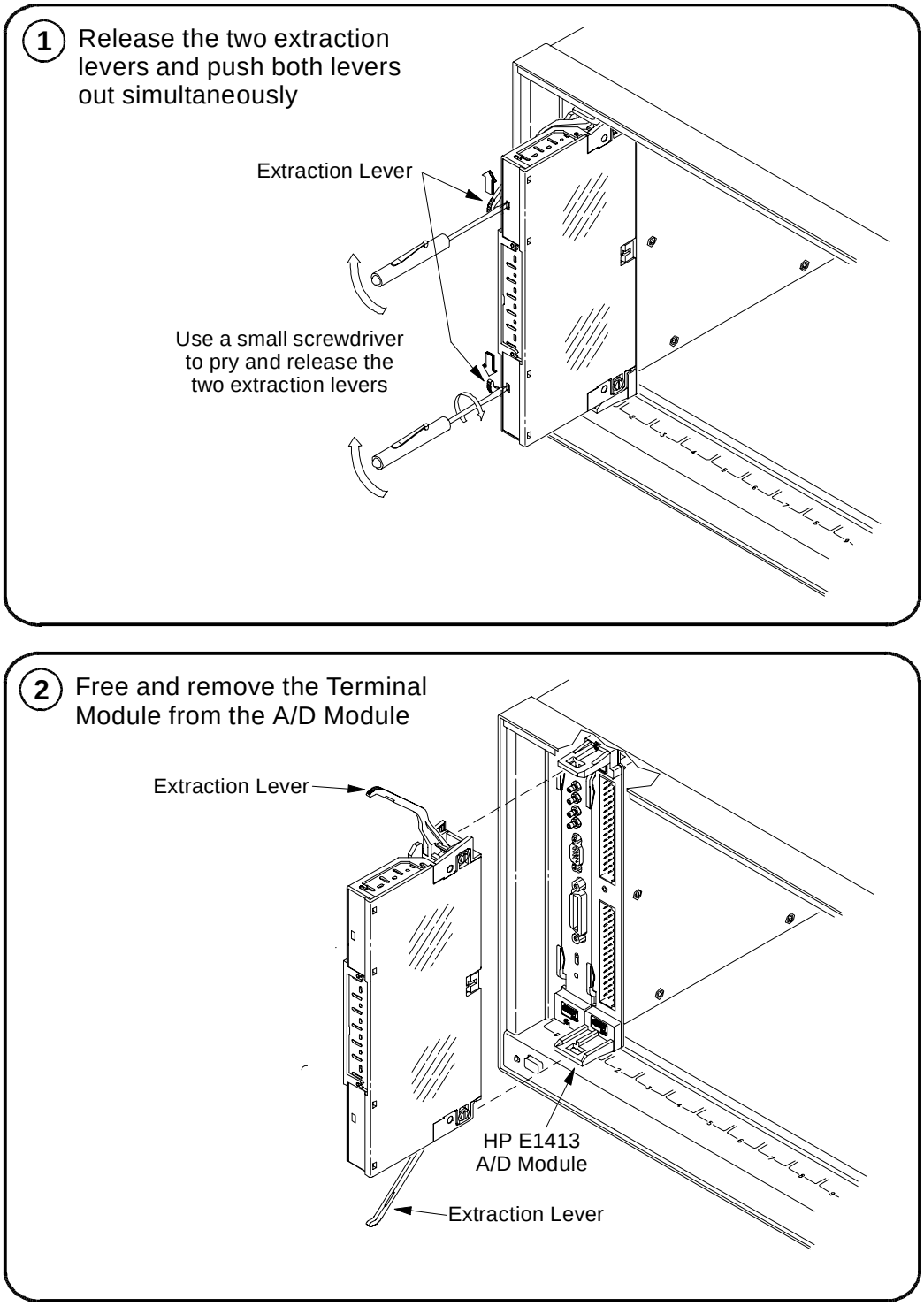


Figure 2-18. Removing the HP E1413 Terminal Module

Terminal Module Wiring Maps

Figure 2-19 shows the terminal module map for the HP E1313 and Figure 2-20 shows the terminal module map for the HP E1413.

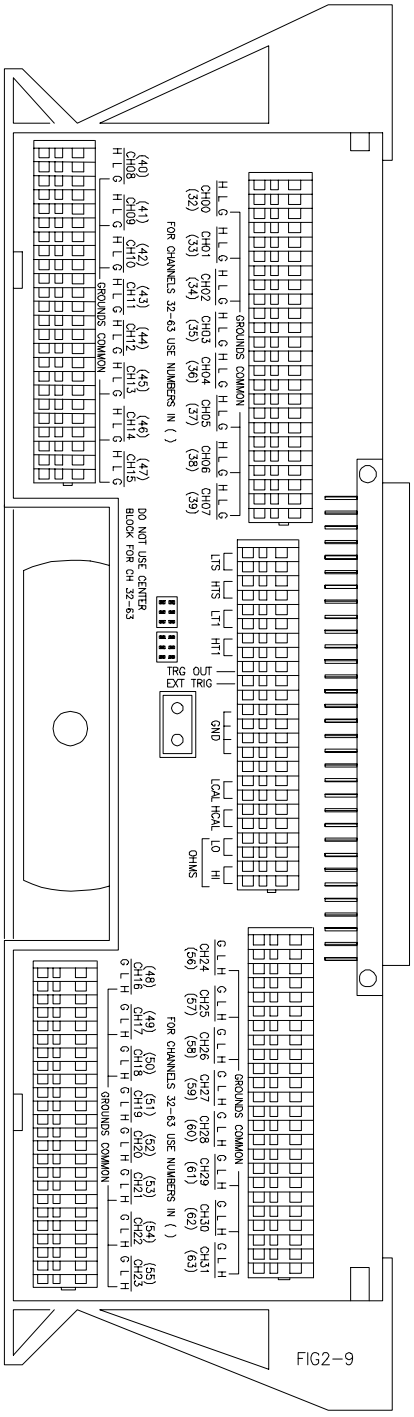


Figure 2-19. HP E1313 Terminal Module Map

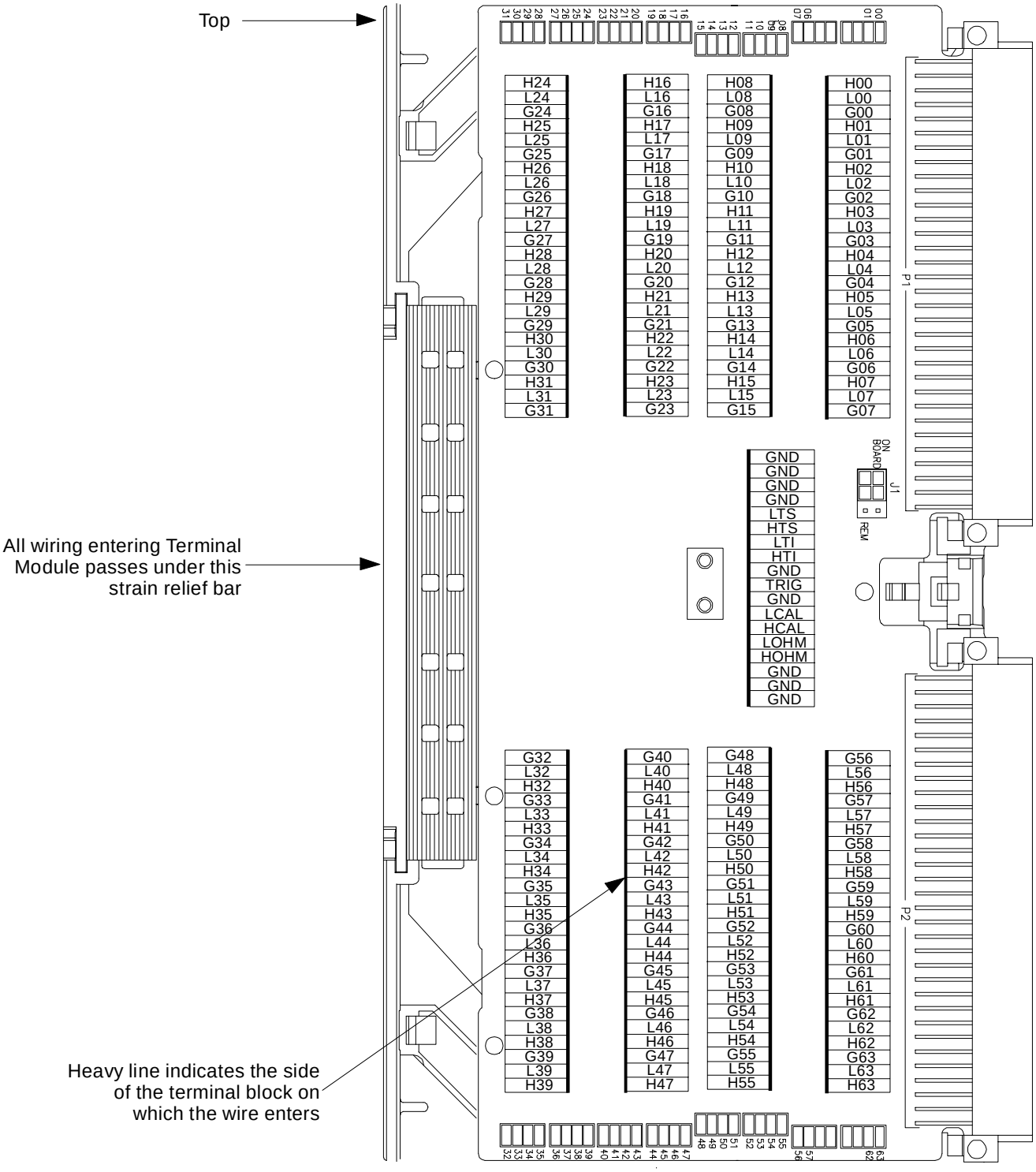


Figure 2-20. HP E1413 Terminal Module Map

Terminal Module Options

Besides the standard terminal module with push-in connectors, the HP E1313/E1413 can be ordered with the following two options. One option (Option A3F) allows connection to an HP E1586 Rack Mount Terminal Panel and the other option (A3E) allows direct connections to the HP E1313/E1413 A/D Module's Faceplate using connectors.

Option A3E

HP E1313/E1413 Option A3E can be ordered if a crimp-and-insert terminal module is desired. This allows you to crimp connectors onto wires which are then inserted directly into the HP E1313/E1413 faceplate connector. Refer to the pin-out diagram on pages 63 and 64 to make the connections. The crimp-and-insert connector is shown in Figure 2-21.

Note The pin numbering on the crimp-and-insert connector may not agree with the pin numbering on the HP E1413's faceplate connector. Use the pin numbering on the faceplate connector to wire the crimp-and-insert connector.

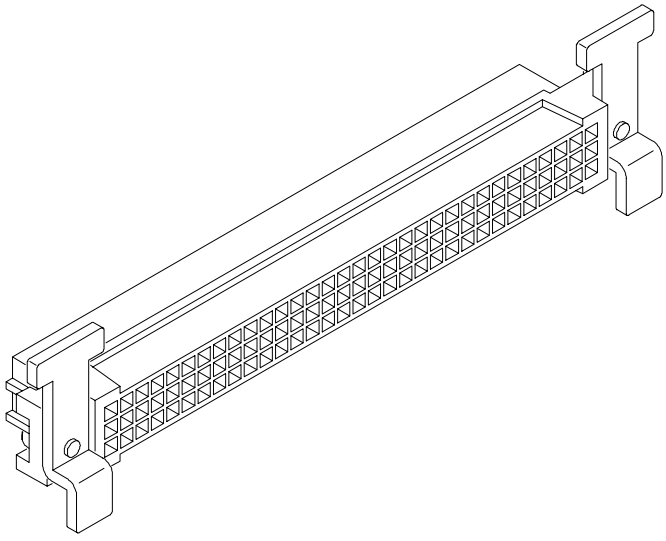


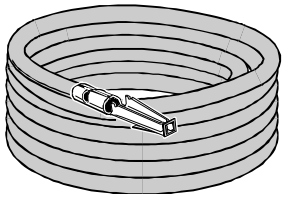
Figure 2-21. Crimp-and-Insert Connector

Crimp-and-Insert Terminal Module Accessories

The following accessories are necessary for use with crimp-and-insert Option A3E:

Single-Conductor and Contact

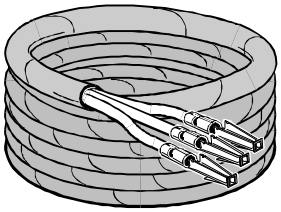
A crimp-and-insert contact is crimped onto one end of a wire. The other end is not terminated. Order HP 91510A.



Length: 2 meters
Wire Gauge: 24 AWG
Quantity: 50 each
Insulation Rating: 105°C maximum
Voltage: 300 V

Shielded-Twisted-Pair and Contacts

A crimp-and-insert contact is crimped onto each conductor at one end of a shielded-twisted-pair cable. The other end is not terminated. Order HP 91511A.



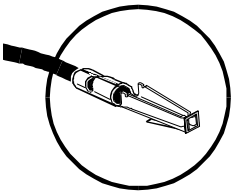
Length: 2 meters
Wire Gauge: 24 AWG
Outside Diameter: 0.1 inches
Quantity: 25 each
Insulation Rating: 250°C maximum
Voltage: 600 V

Jumper Wire and Contacts

A crimp-and-insert contact is crimped onto each end of a single-conductor jumper wire. This jumper is typically used to tie two pins together in a single crimp-and-insert connector. Order HP 91512A.

Crimp-and-Insert Contacts

These contacts may be crimped onto a conductor and then inserted into a crimp-and-insert connector. The crimp tool kit is required to crimp the contacts onto a conductor and remove the contact from the connector. Order HP 91515A.



Wire Gauge Range: 20 - 24 AWG
Quantity: 250 each
Plating: Gold Plated Contact
Maximum Current: 2A at 70°C

Crimp-and-Insert Tools

The hand crimp tool (part number HP 91518A) is used for crimping contacts onto a conductor. The pin extractor tool (part number HP 91519A) is required for removing contacts from the crimp-and-insert connector. **These products are not included with Option A3E or with the terminal option accessories listed earlier.**

Extra Crimp-and-Insert Connectors

The crimp-and-insert connector is normally supplied with Option A3E. Contact Hewlett-Packard Company if additional connectors are needed. Order HP 91484B.

Option A3F

Option A3F shown in Figure 2-22 allows an HP E1413/E1313 to be connected to an HP E1586A Rack Mount Terminal Panel. The A3F option for the HP E1413 provides 4 SCSI plugs on a Terminal Module and the option for the HP E1313 provides 2 SCSI plugs on a Terminal Module. For HP E1313s with 64-channels, use an additional Option A3F Terminal Module.

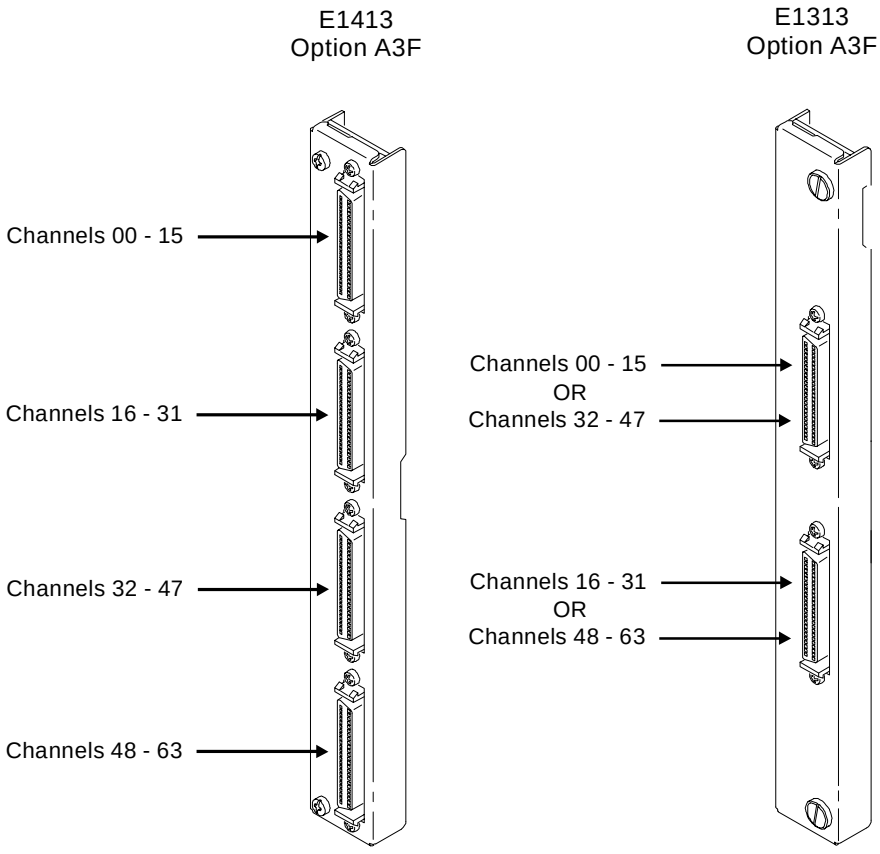


Figure 2-22. HP E1413 Option A3F

The Option A3F Terminal Module enables connections to an HP E1586A Rack Mount Terminal Panel. Note that to connect all 64-channels, two HP E1586A Rack Mount Terminal Panels are required. Figure 2-25 shows how to connect the Terminal Module to the HP E1413 A/D Module. See "Connecting and Mounting the HP E1586A Rack Mount Terminal Panel" on page 61 for information on connecting the HP E1586A Rack Mount Terminal Panel.

Option A3F Pin-Out and Signal Lines

Figure 2-23 shows the pin-out and signal lines for the HP E1313 Option A3F Terminal Module, and Figure 2-24 shows the pin-out and signal lines for the HP E1413 Option A3F Terminal Module.

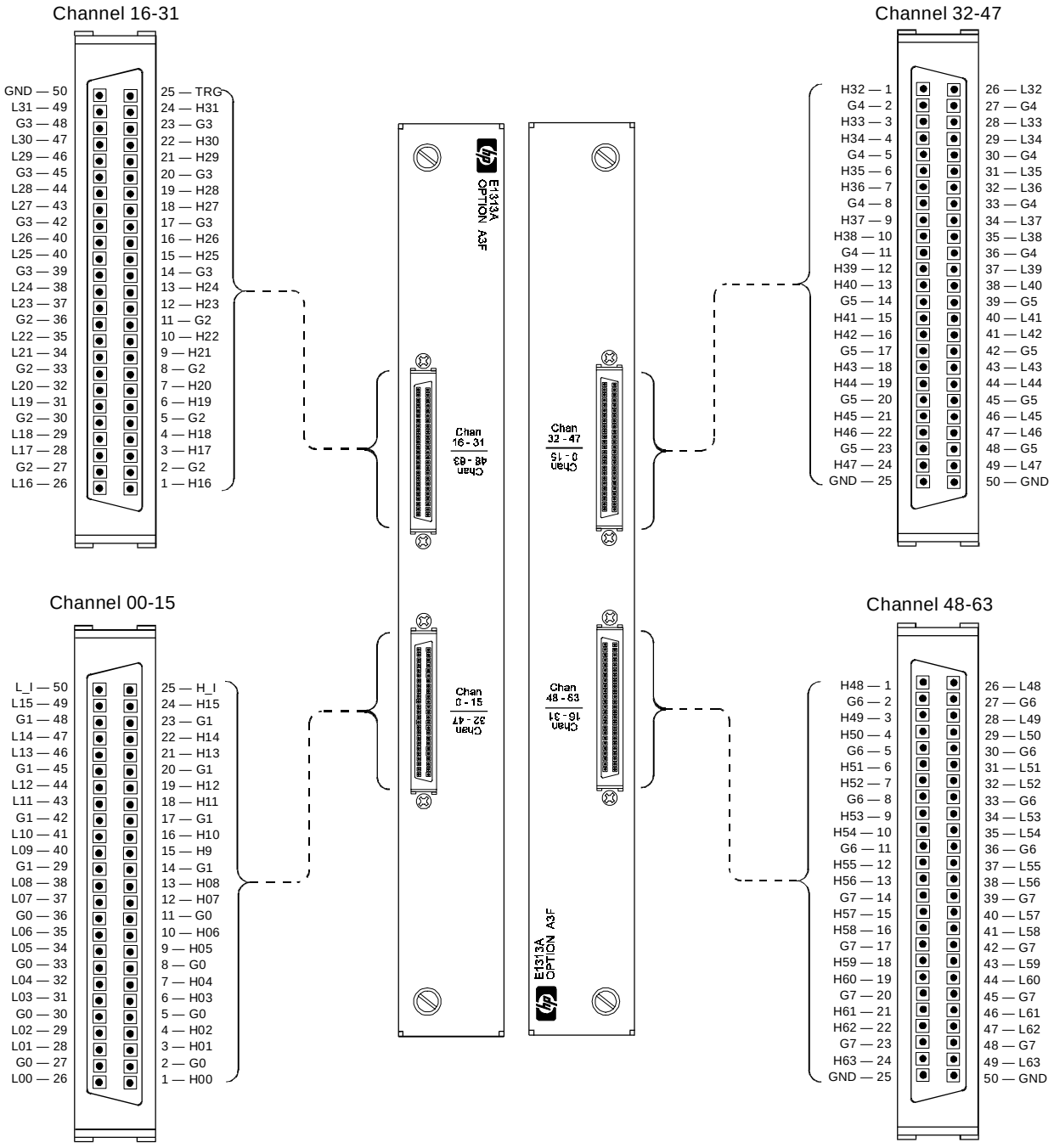


Figure 2-23. HP E1313 Option A3F Pin-out

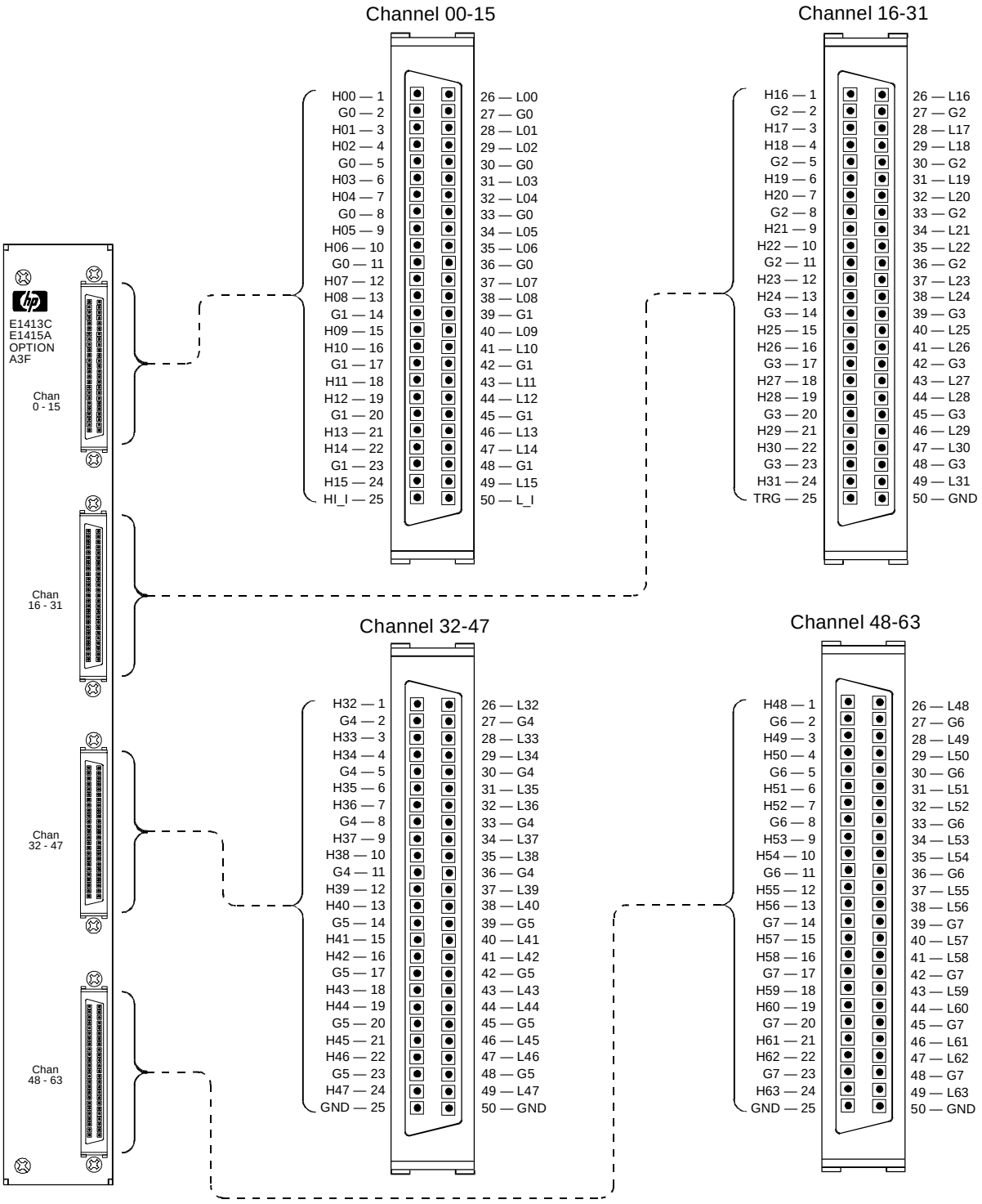


Figure 2-24. HP E1413 Option A3F Pin-out

Connecting and Mounting the HP E1586A Rack Mount Terminal Panel

The HP E1586 Rack Mount Terminal Panel provides extended connections to the HP E1313/E1413 modules. The Terminal Panel is recommended if the HP E1313/E1413 is located a distance away from the measurement connections. The Terminal Panel provides up to 32, 3-wire connections to allow for 32 channel connections to the A/D modules. See "Using the HP E1586A Rack Mount Terminal Panel" on page 93 for operating information.

Connecting the HP E1586A Rack Mount Terminal Panel

Figure 2-25 shows how to connect the HP E1586A Rack Mount Terminal Panel to the HP E1413.

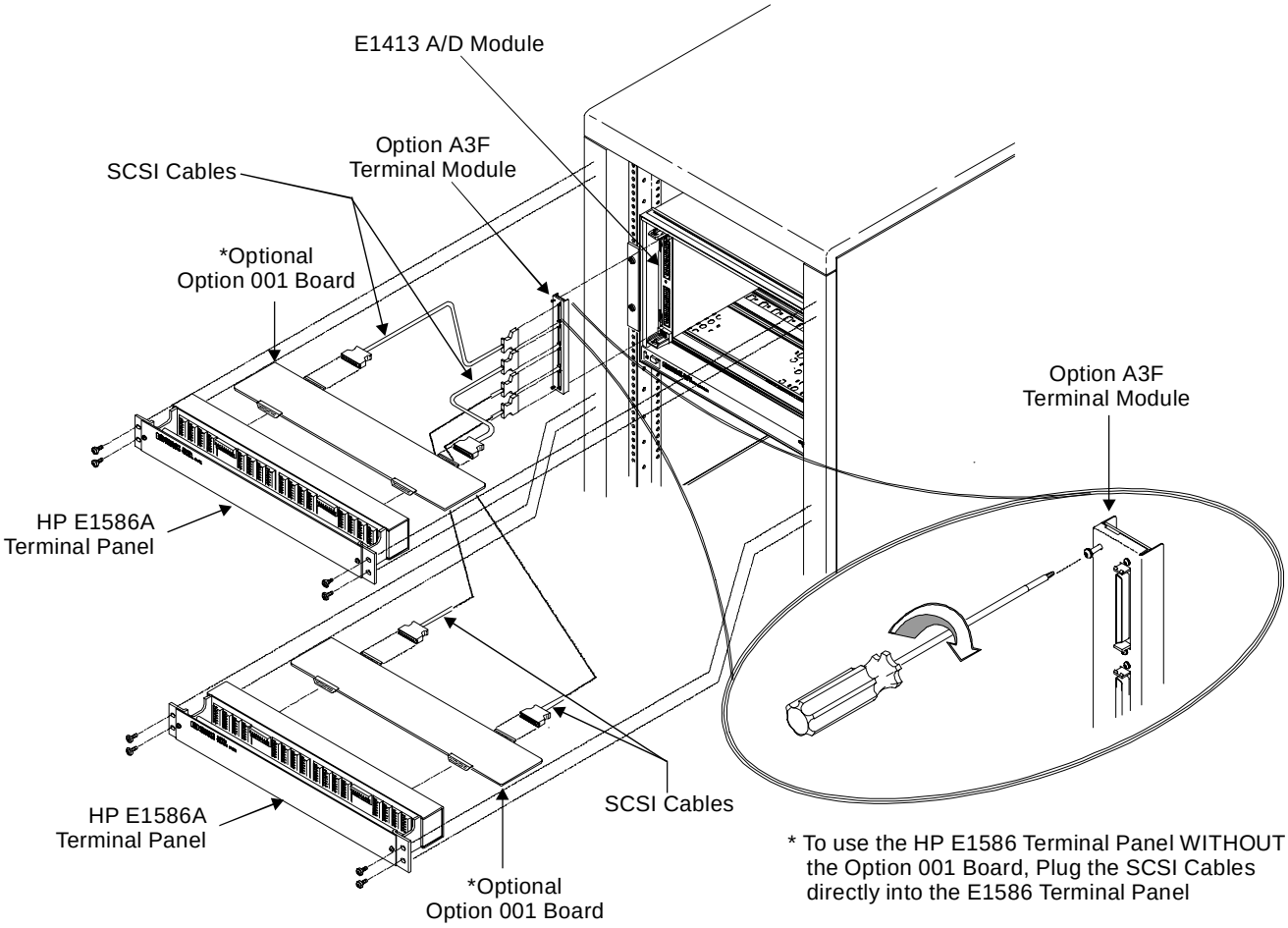


Figure 2-25. Connecting the HP E1586A Rack Mount Terminal Panel

Interconnect Cables

There are two different cables available for connecting the HP E1586 Rack Mount Terminal Panel to the HP E1313/E1413 Option A3F. In both cases, four cables are required if all 64-channels are needed. This also requires two HP E1586A Rack Mount Terminal Panels since a single Terminal Panel only connects 32 channels. The cables do not come with the HP E1313/E1413 Option A3F and must be ordered separately. The cables are described in the following paragraphs.

Standard Cable

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

Custom Length Cable

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

Mounting the HP E1586A Rack Mount Terminal Panel

The Terminal Panel can be mounted in a standard size instrument rack. To minimize temperature gradients across the panel, it should be mounted in the rack such that it is away from the other heat sources. The bottom of the rack is usually the preferred location. Take particular care to minimize the temperature differences across the horizontal width of the Terminal Panel, since it is most susceptible to horizontal temperature gradients across its longest dimension.

HP E1586A Option 001 HF Common Mode Filters

Optional high frequency common mode filters on the HP E1586A Rack Mount Terminal Panel's input channels filter out AC common mode signals present in the cable that connects the terminal panel and the device under test. These filters are useful for filtering out small common mode signals below 5Vp-p. To order these filters, order HP E1586A Option 001.

Faceplate Connector Pin-Signal Lists

Figure 2-26 shows the faceplate connector pin-signal list for the HP E1313 and Figure 2-27 shows the list for the HP E1413.

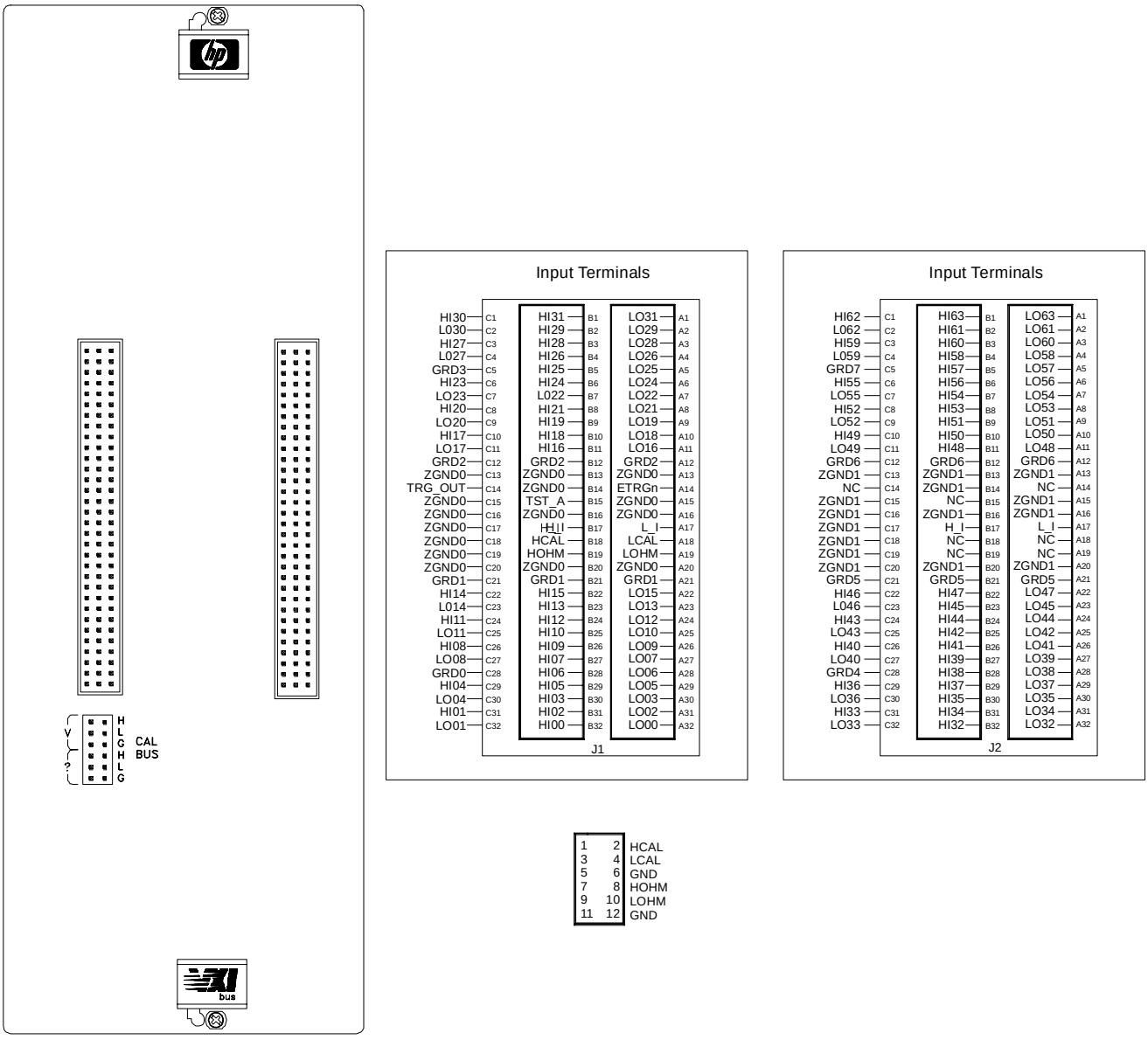
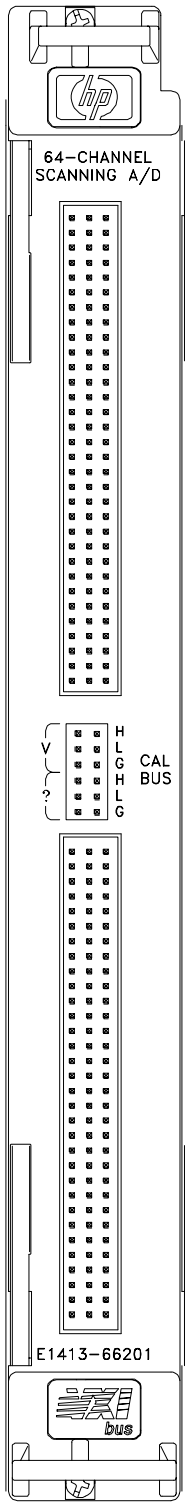


Figure 2-26. HP E1313 Connector Pin-Signal List



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

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

1	GND	GND	GND	32
2	GND	TRG	GND	31
3	GND	GND	GND	30
4	TST-A	SYSF	GND	29
5	GND	GND	GND	28
6	G4	G4	G4	27
7	G4	G4	G4	26
8	G4	G4	G4	25
9	L33	H32	L32	24
10	H33	H34	L34	23
11	L36	H35	L35	22
12	H36	H37	L37	21
13	G5	H38	L38	20
14	L40	H39	L39	19
15	H40	H41	L41	18
16	L43	H42	L42	17
17	H43	H44	L44	16
18	L46	H45	L45	15
19	H46	H47	L47	14
20	L49	H48	L48	13
21	H49	H50	L50	12
22	L52	H51	L51	11
23	HI52	H53	L53	10
24	L55	H54	L54	9
25	H55	H56	L56	8
26	G6	H57	L57	7
27	G7	H58	L58	6
28	GND	H59	L59	5
29	GND	H60	L60	4
30	GND	H61	L61	3
31	GND	H62	L62	2
32	GND	H63	L63	1

Figure 2-27. HP E1413 Connector Pin-Signal List

Chapter 3
Using the HP E1313/E1413

About This Chapter

Except where noted, all references to the HP E1413 apply to the HP E1313. This chapter introduces programming the HP E1413 with the SCPI instrument language. Chapter contents include:

Module Description

The HP E1313/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 autoranging 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 E1313/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 E1313/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.

Default Settings After Power-on, *RST, or *TST?

Before discussing the ten programming steps and their recommended sequence, it is important to understand that for a given programming situation, you may not have to send commands to set conditions that are default states. The default instrument states are:

SCP Set-up:	All SCP settings defaulted (see SCP manuals).
A/D Filter:	OFF
Engineering Units (EU) linked to channels:	DC Volts, autorange linked to channels 0 through 63.
Scan list assignment and selection:	Scan list 1 is selected and has channels 0 through 63 assigned.
Trigger system:	Trigger system is in Trigger Idle State (not INITiated). Arm source set to IMMediate. Trigger source set to HOLD. Trigger count set to 1. Sample Timer set to 10 ms.
Data Format:	ASCII
FIFO mode:	BLOCK

Default Operation

By executing only three commands after *RST, the HP E1313/E1413 will execute a single scan of all 64 channels and return 64 voltage values from the FIFO.

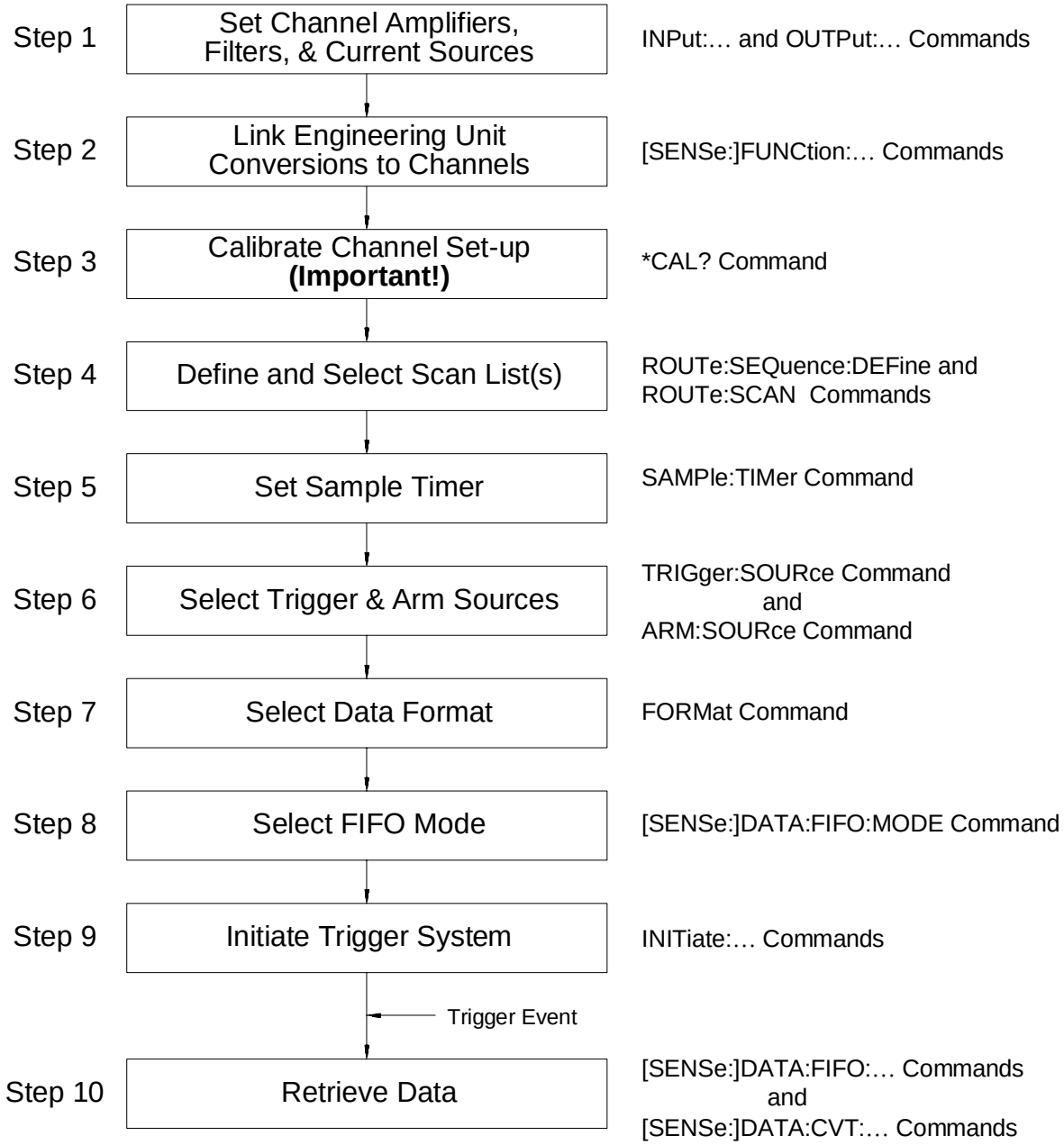
Example Command Sequence

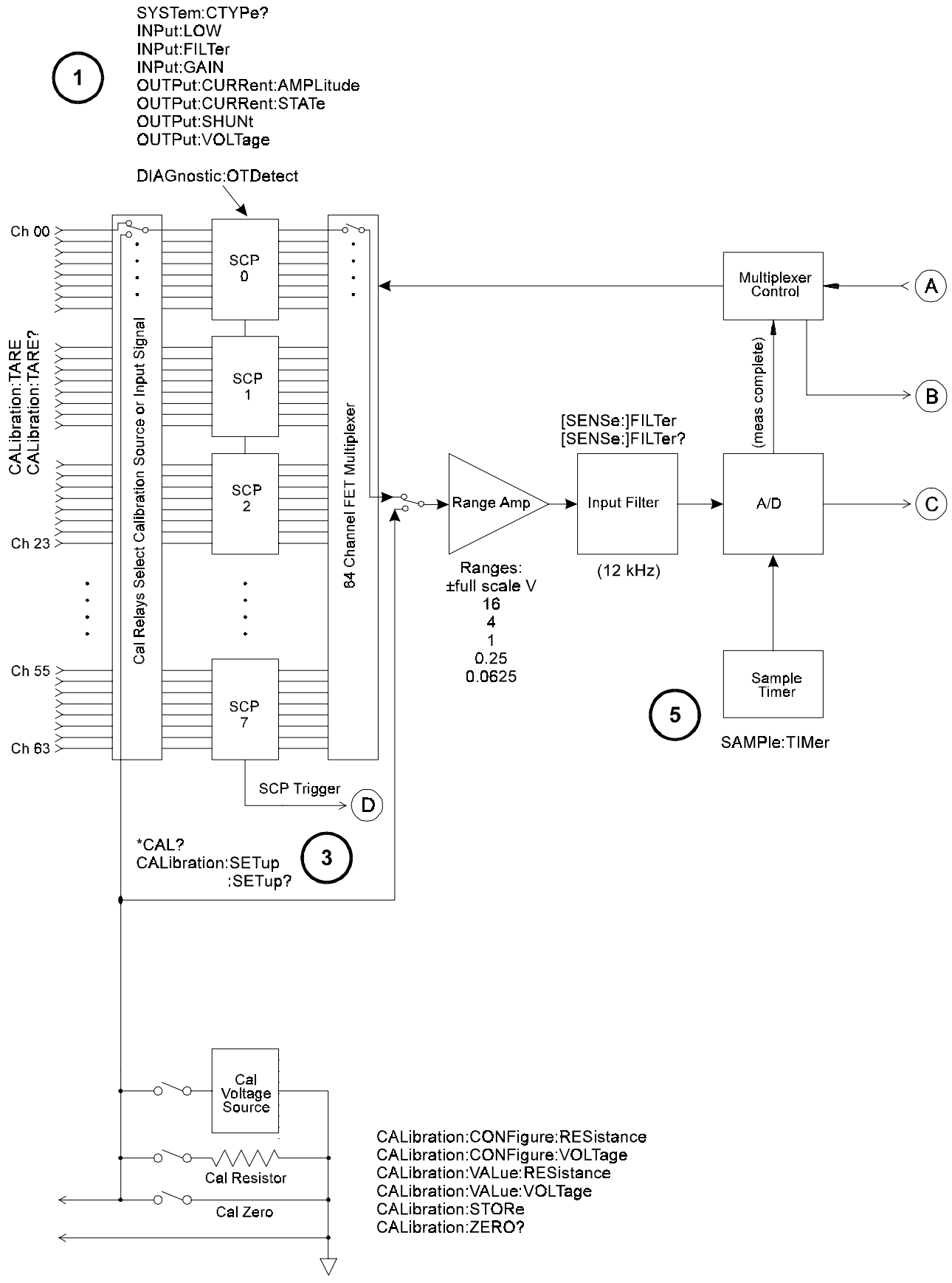
```
*RST                        Insure module is in default state.
INIT:IMM                    Initiate trigger system for scan.
TRIG:IMM                    Software trigger begins scan.
SENS:DATA:FIFO:ALL?         Recover 64 readings from FIFO.
```

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 on pages 68 and 69.

Important! It is very important while developing your application that you execute the SYSTem:ERRor? command after each programming command. This is the only way you will know if there is a programming error.





Continued on Next Page

Figure 3-1. SCPI Programming Overview

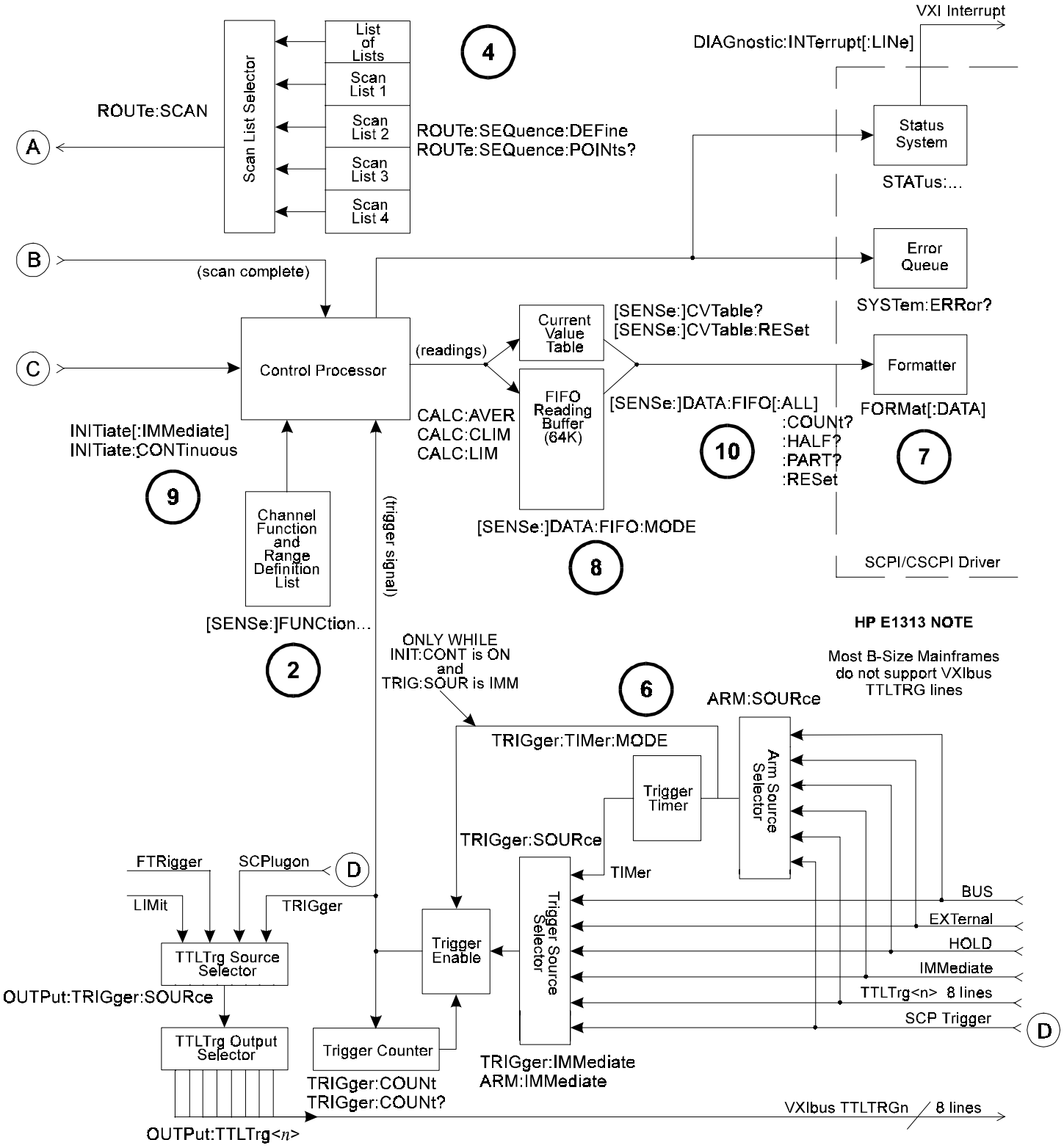


Figure 3-1 (continued). SCPI Programming Overview

Step 1. Setting-up Signal Conditioning Plug-ons

This step does not apply to non-programmable Signal Conditioning Plug-ons such as the HP E1501 Direct Input SCP or the HP E1502 Low Pass Filter SCP.

Setting SCP Gains

The gain command for programmable amplifiers is
INPut:GAIN <*gain*>,(@<*ch_list*>)

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 HP E1503 Programmable Filter/Gain 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 autorange. Autorange 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.

and

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

or to combine into a single command message

```
INP:FILT:LPAS: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 INPut:FILTer[:LPASs][:STATe] ON command. For example, to program all but a few filters on, send:

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

```
INP:FILT:LPAS: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 <*amplitude*>,(@<*ch_list*>) and OUTPut:CURRent[:STATe] <*enable*>.

- The <*amplitude*> parameter sets the current output level. It is specified in units of ADC and for the HP E1505 Current Source 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:AMPL  30e-6,(@100:109)
OUTP:CURR:AMPL  488e-6,(@110:119)      Separate command per output level
```

or to combine into a single command message:

```
OUTP:CURR:AMPL  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 versus channel numbers.

The commands for linking EU conversion to channels are:

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

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

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

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

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

Notes

1. At power-on and after *RST, the default EU conversion is autorange voltage for all 64 channels.
2. While the A/D filter is on ([SENSe:]FILTer[:LPASs][:STATe] ON), <*range*> may not specify autorange. The filtered settling time would be too great for autorange to function. Use SCP filtering (filtering at each channel) to allow use of autorange at high speed.

Linking Voltage Measurements

To link channels to the voltage conversion send the [SENSe:]FUNCtion:VOLTage[:DC] [<*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 autorange.

To set channels 0 through 15 to measure voltage using autorange:

```
SENS:FUNC:VOLT:DC  0,(@100:115)        0 for range means autorange.
```

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

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

or to send both commands in a single command message:

```
SENS:FUNC:VOLT:DC  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 autorange. Autorange 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 3-2.

- 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 HP E1505 Current Source 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 autorange.

- 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 3-2 for diagrams of these measurement connections.

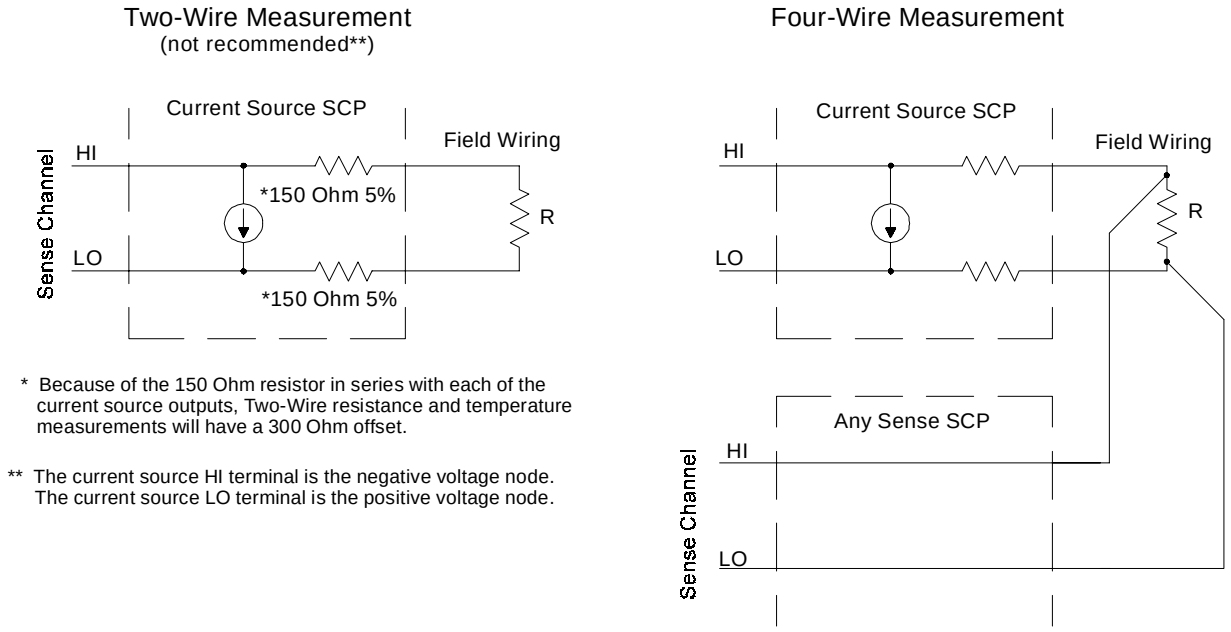


Figure 3-2. Resistance Measurement Sensing

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 *<sensor_type>*,*<sub_type>*,[*<range>*,](@*<ch_list>*) command.

- The *<ch_list>* parameter specifies which channel(s) to link to the temperature EU conversion.
- The *<sensor_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 autorange.

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 3-2 in "Linking Resistance Measurements".

For resistance temperature measurements the *<sub_type>* parameter specifies:

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

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 paired 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 "Setting SCP Gains" on page 70, 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 (CUSTomis predefined 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 <*sensor_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 <*sensor_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 0.003**85** or 0.003**92** Ohms/Ohm/Degree C temperature coefficients, respectively).
  - For Thermistors; only 5000 (See previous note on page 75).
  - 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 autorange.

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 Sensing with the HP E1313" on page 40 or "Reference Temperature Sensing with the HP E1413" on page 41 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 E1506 and HP E1507). To link channels to strain EU conversions send the [SENSe:]FUNCtion:STRain:<*bridge_type*> [<*range*>,](@<*ch_list*>) command.

- <*bridge_type*> is not a parameter, but is part of the command syntax. The following table relates the command syntax to bridge type. See the *HP E1506/E1507 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
: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 <u>not</u> 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 autorange.

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 Chapter 5 and the *HP E1506/E1507 User's 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 HP 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 HP E1413's memory for each channel's EU conversion table. When standard EU conversions are specified, the HP 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 (EU) 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 channels 32-63 to HP E1413.
SENS:FUNC:CUST 1,1,(@132:163)             Link custom EU with channels 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 EU 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 channels 24-31 to HP E1413.
SENS:FUNC:CUST  1,1,(@124:131)             Link custom EU with channels 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 channels with the custom EU conversion table loaded with the DIAGnostic:CUSTom:LINear or DIAGnostic:CUSTom:PIECewise commands.

Two other custom linking commands are available for custom thermocouple measurements:

[SENSe:]FUNCtion:CUSTom:TCouple <*type*>,[<*range*>,](@<*ch_list*>)

and

[SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>)

A full explanation of these custom temperature EU linking commands can be found starting on page 136.

- The <*range*> parameter selects one of the HP 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 autorange. The default range (no range parameter specified) is autorange. If an A/D reading is greater than the <*table_range*> specified with DIAGnostic:CUSTom:PIECewise, an overrange condition will occur.
- The <*type*> command is for the SENSe:FUNCtion:CUSTom:TCouple command and it specifies the type of thermocouple wire.
- If no custom table has been loaded for the channels specified with SENSe:FUNCtion:CUSTom, an error will be generated when an INITiate 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
                                          channels 16-23.
SENS:FUNC:CUST  1,(@124:131)              Link custom EU with channels
                                          24-31 and set the 1V A/D range.
INITiate then TRIGger module
```

Step 3. Performing Channel Calibration (Important!)

*CAL? (also performed using CAL:SETup then CAL:SETup?) is a very important step. *CAL? generates calibration correction constants for all analog input and output channels. *CAL? must be performed in order for the HP E1313/E1413 to deliver its specified accuracy.

Operation and Restrictions

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

For analog measurement excitation SCPs (such as the HP E1505) *CAL? also generates calibration correction constants. These calibration constants are valid only for the specific SCPs in the positions they are currently in. The calibration becomes invalid if you move these SCPs to different SCP locations.

How to Use *CAL?

When you turn power on to the HP E1313/E1413 after you have first installed your SCPs (or after you have moved SCPs), the module will use approximate values for calibration constants. This means that input and output channels will function although the values will not be very accurate relative to the HP E1313/E1413's specified capability. At this point, make sure the module is firmly anchored to the mainframe (front panel screws are tight), and let it warm up for a full hour. After it has warmed up, execute *CAL?.

What *CAL? Does

The *CAL? command causes the module to calibrate A/D offset and gain, and all channel offsets. This may take many minutes to complete. The actual time it will take your HP E1313/E1413 to complete *CAL? depends on the mix of SCPs installed. *CAL? performs literally hundreds of measurements of the internal calibration sources for each channel and must allow 17 time constants of settling wait each time a filtered channel's calibration source changes value. The *CAL? procedure is internally very sophisticated and results in an extremely well calibrated module.

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

When to Re-Execute *CAL?

- When you change the channel gain and/or filter cut-off frequency on programmable SCPs (using INPut:GAIN or INPut:FILTer:… commands).
- When you reconfigure SCPs to different locations. This is true even if you replace an SCP with an identical model SCP because the calibration constants are specific to each SCP channel's individual performance.
- When the ambient temperature within the mainframe changes significantly. Temperature changes affect accuracy much more than long-term component drift.

Note To save time when performing channel calibration on multiple HP E1313/E1413s in the same mainframe, use the CAL:SETup and CAL:SETup? commands (see these commands in Chapter 5 for details).

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 E1313/E1413 you can define up to 4 different scan lists each with its own channel-to-channel sample pacing (see "Step 5. Setting the Sample Timer"). You can also define a List-of-Lists (automatic scan list sequencing) by defining LISTL to contain a list of scan lists. 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, ALL or LISTL. When *scan_list* specifies LISTL, *ch_list* must specify scan lists rather than channels. This builds a list of scan lists to execute.
- The <*ch_list*> parameter must specify at least 2 channels (3 channels when an HP E1511 or HP E1512 SCP is installed) except when the scan list will be specified by LISTL. In this case the scan list must contain at least 6 channels. The channels can be specified in any order and the same channel can be specified more than once. A scan list can contain as many as 1,024 channel specifiers.
- When ROUTe:SEQuence:DEFine 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 ROUTe:SEQuence:DEFine command.
- After a *RST command or at power-on, channel list 1 (LIST1) is predefined as ROUTe:SEQuence:DEFine LIST1,(@100:163). All other scan lists are undefined.

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, LIST4, or LISTL. You define LISTL as a list of scan lists when LISTL is the current *scan list*.
- ROUTe: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 ROUTe:SCAN is executed after this point, it will become effective for the next scan. If INITiate:CONTinuous is ON and TRIGger:SOURce is IMMediate, ROUTe:SCAN will generate an error. See Figure 3-3.

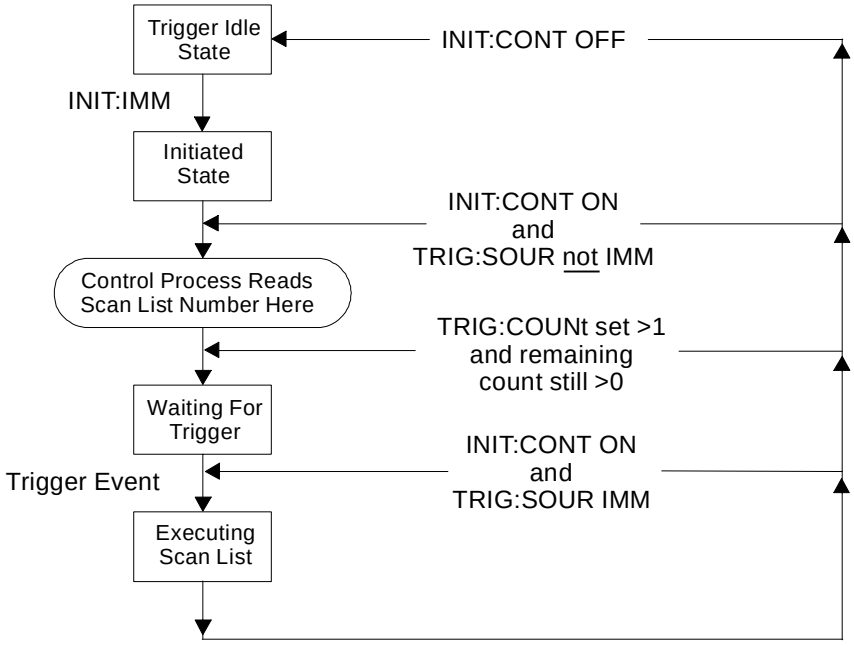


Figure 3-3. Event Sequence for ROUTe:SCAN

To select scan list 2 as the current scan list:

```
ROUT:SCAN  LIST2
```

Current scan list becomes 2 when module initiated to wait-for-trigger.

Step 5. Setting the Sample Timer

The SAMPle:TIMer <*scan_list*>,<*interval*> command individually programs channel-to-channel sample pacing. The power-on default allows maximum sample rate.

- The <*scan_list*> parameter can specify LIST1, LIST2, LIST3, or LIST4 or LISTL. When <*scan_list*> is LISTL, the sample interval will apply to all scan lists defined for LISTL while ROUTe:SCAN selects LISTL.
- The <*interval*> parameter can specify 10.0e-6 (MIN) to 32.7680e-3 (MAX) seconds with a resolution of 0.5e-6 seconds.

Note While the A/D filter is on ([SENSe:]FILTer[:LPASs][:STATe] ON) the minimum sample time is 145µsec. Use SCP filtering (filtering at channels) to allow scanning at high speed.

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 3-4 shows the trigger and arm model for the HP E1413. Note that when the trigger source selected is TIMer, or IMM with INITiate:CONTinuous ON, the remaining sources become arm sources. Using ARM:SOURce allows you to specify an event that must occur in order to start scanning.

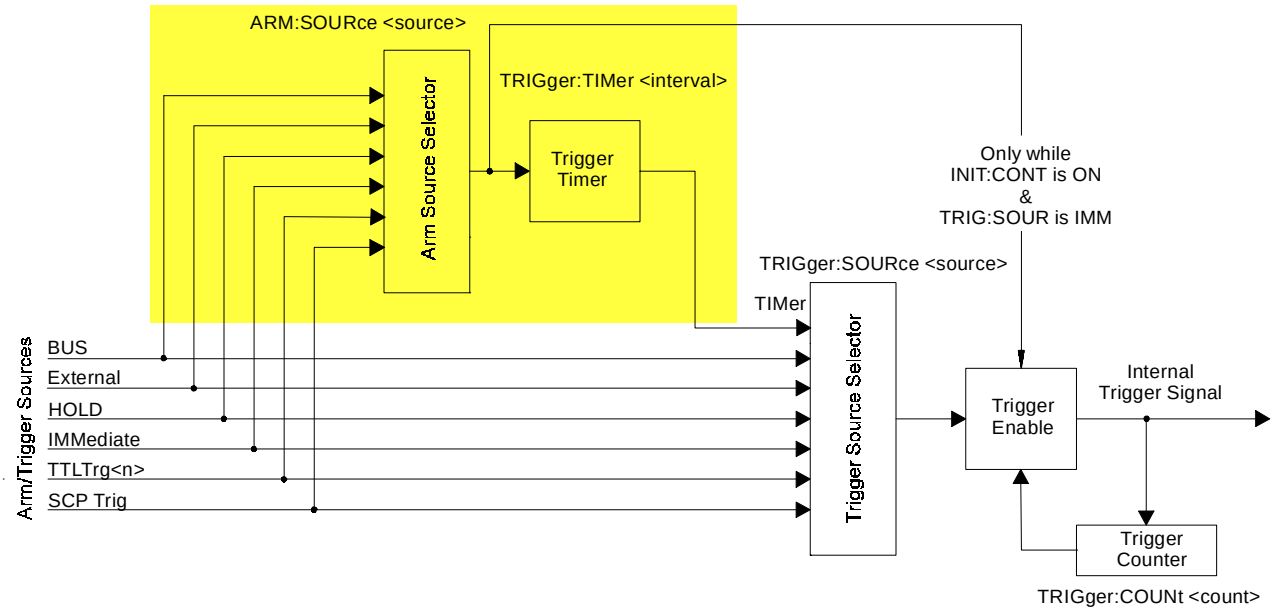


Figure 3-4. Logical Arm and Trigger Model

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 possible choices for the *<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 or IMMediate with INITiate:CONTinuous ON, ARM:SOURce must be set to IMMediate (the *RST condition). If not, the INITiate command will generate Error -221, "Settings conflict".
2. When TRIGger:SOURce is TIMer, the trigger timer interval (TRIGger:TIMer <*interval*>) must allow enough time to scan all channels (ROUTe:SEQuence:DEFine) in the active scan list (ROUTe:SCAN) or a +3012, "Trigger Too Fast" error will be generated during the measurement scan. See the Command Reference for TRIGger:TIMer[:PERiod] or SAMPle:TIMer for timing details.

To set the Trigger Source to a software trigger (TRIGger:IMMediate, 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 TIM                     Now select ARM:SOURce.
```

Selecting Timer and Continuous Mode Arm Source

Figure 3-4 shows that when the TRIGger:SOURce is TIMer, or IMMediate with INITiate:CONTinuous ON, the other trigger sources become arm sources and control when the timer will start or scanning will begin. The command to select the 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)

Note

When TRIGger:SOURce is not TIMer, or IMMediate with INITiate:CONTinuous ON, ARM:SOURce must be set to IMMediate (the *RST condition). If not, the INITiate command will generate 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 TRIGger:COUNt is set to INFinite, the trigger counter is disabled and the module will accept an unlimited number of trigger events.

To set the trigger counter to 100:

```
TRIG:COUN  100
```

Can trigger 100 times after a single INITiate 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:DATA <*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 the 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:

```
FORM:DATA  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:

```
FORM:DATA  ASC,7
```

*The *RST, *TST? and power-on default format.*

or

```
FORM:DATA  ASC
```

Step 8. Selecting the FIFO Mode

The HP E1313/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 STATus:QUEStionable:CONDition? bit 10 (FIFO Overflowed), and an error message is put in the error queue (read with SYSTem:ERRor? command).

- **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):

```
SENS:DATA:FIFO:MODE  BLOCK        Select blocking mode.
SENS:DATA:FIFO:MODE  OVER         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 E1313/E1413 from the trigger idle state to the wait-for-trigger state. When initiated, the instrument is ready to receive one (INITiate[:IMMediate]) or more (INITiate:CONTinuous ON) trigger events.

To initiate the instrument for the number of triggers specified with the TRIGger:COUNt command:

```
INIT      or      INIT:IMM
```

To have the instrument continuously in the wait-for-trigger state:

```
INITiate:CONTinuous  ON
```

If INITiate:CONTinuous is ON and TRIGger:SOURce is IMMediate, the module scans continuously until you execute INITiate:CONTinuous OFF.

When an INITiate 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 SYSTem:ERRor? command). Some examples:

- If TRIGger:SOURce is not TIMer or IMMediate with INITiate:CONTinuous ON, then ARM:SOURce 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 Current Value Table:

[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[:DATA] 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:

SENS: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 will 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 is not the only FIFO data retrieval command. There are four other commands and they are covered in " Advanced FIFO Data Retrieval" beginning on page 101.

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*) × (*number of scans*)

The *number of channels* is determined by the ROUTe:SEQuence: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 INITiate:CONTinuous is ON, the number of triggers generated by TRIGger:SOURce and the specified TRIGger:COUNt control the *number of scans.*

To transfer 1,024 readings from the instrument:

SENS: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.*

Example Program

An example C-SCPI program (*using.cs*) is located on the C-SCPI driver tape, and example C and Visual BASIC program (*using.xx*) is on the driver/example program disk. These programs put 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.

```
*RST                                    Reset the module.
    Setting up Signal Conditioning (step 1 only for programmable SCPs).
INP:FILT:LPAS:FREQ  2,(@116:119)
INP:GAIN  64,(@116:119)
INP:GAIN  8,(@120:123)
    Link channels to EU conversions (step 2).
SENS:FUNC:VOLT  AUTO,(@100:107)
SENS:REF:TEMP  THER,5000,AUTO,(@108)
SENS:FUNC:TEMP  TC,T,AUTO,(@109:123)
    Execute channel calibration (step 3).
*CAL?
    Define and select scan list (step 4).
ROUT:SEQ:DEF  LIST1,(@108:123,100:107)
ROUT:SCAN  LIST1                        This is a *RST default.
    Set sample timer (step 5).
SAMP:TIM  LIST1,.00001
    Configure the trigger system (step 6).
ARM:SOUR  IMM                           This is a *RST default.
TRIG:COUN  10
TRIG:TIM  .0007
TRIG:SOUR  TIM
    Specify data format (step 7).
FORM:DATA  REAL,32
    Select FIFO mode (step 8).
SENS:DATA:FIFO:MODE  BLOCK
    Initiate trigger system (step 9).
INIT:CONT  OFF                          This is a *RST default.
INIT:IMM
    Retrieve data (step 10).
SENS:DATA:FIFO:PART?  <read_data>
```

Using the HP E1586A Rack Mount Terminal Panel

This section shows how to use the HP E1313/E1413 with the HP E1586A Rack Mount Terminal Panel.

Thermistor Connections and Operations

The HP E1586 Rack Mount Terminal Panel's three thermistors are located next to the channel 3 terminal block, between channels 11 and 16, and next to channel 24 (see Figure 3-5). The following section shows how to connect and measure the thermistors.

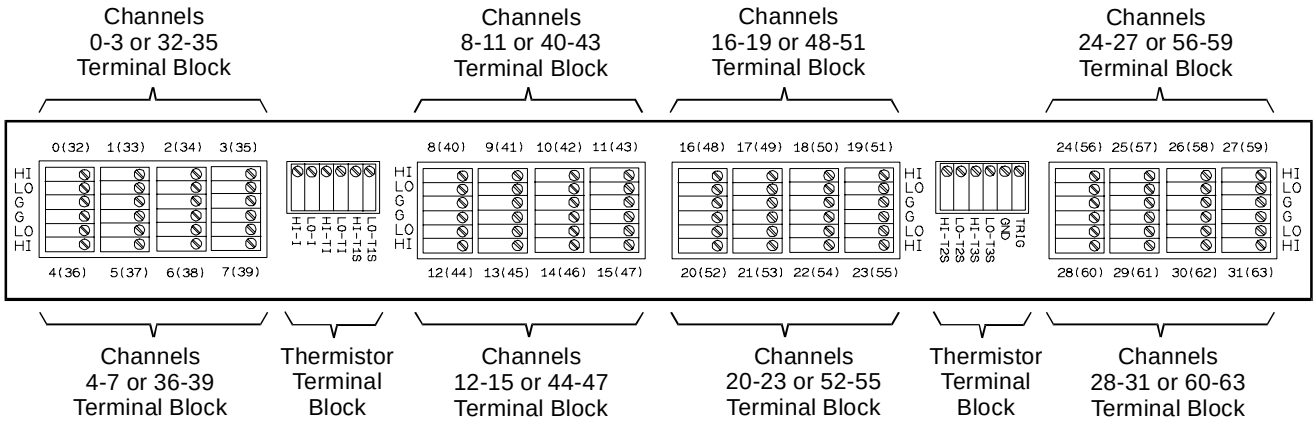


Figure 3-5. Terminal Panel Connections

Thermistor Excitation Sources

The HP E1313/E1413 provides a 122µA current source as the excitation for the thermistors. This is available on the Terminal Panel's terminals labeled HI-I and LO-I. The excitation current is ONLY available to the Terminal panel connected to channels 00 - 31 of the HP E1413. This current is NOT on the Terminal Panel connected to channels 32 - 63 of the HP E1413.

The following section shows how to connect the thermistors to the HP E1413 current source. For the Terminal Panel's on-board thermistors excitation, connect HI-I to HI-TI, and LO-I to LO-TI, respectively.

Reference Temperature Measurements

This section shows how to make reference temperature measurements on the HP E1586 thermistors using the HP E1313/E1413 module. The methods to measure the thermistor reference temperature depends on the location of the Terminal Panel. For Terminal Panels mounted away from heat sources, it is only necessary to measure the center thermistor. Use the information in "Measuring Using the Center Thermistor".

For Terminal Panels mounted in such a way that temperature gradients are generated along its length, measure all three thermistors. Use the information in "Measuring Using the Left, Center, and Right Thermistors".

Connecting One Terminal Panel for Reference Temperature Measurements

In this configuration, a single Terminal Panel is used with an HP E1413 to provide up to 32 channels for temperature measurements. Make the following connections on the thermistor terminal block of the Terminal Panel:

- Connect HI-I to HI-TI
- Connect LO-I to LO-TI

This provides the excitation current to all three on-board thermistors on the Terminal Panel. Figure 3-6 shows the connection for a single Terminal Panel.

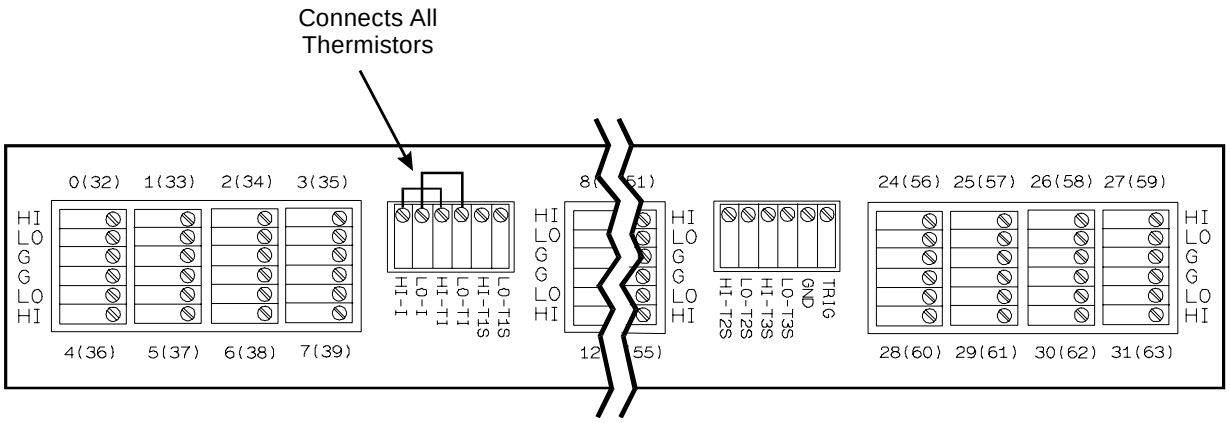


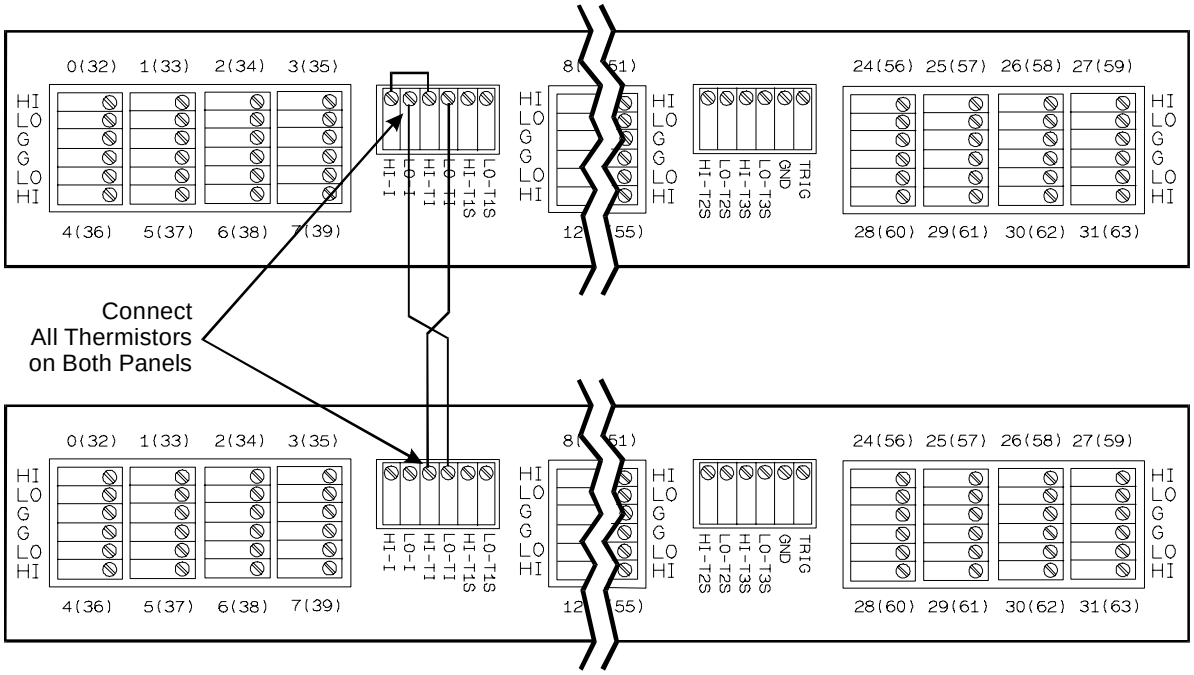
Figure 3-6. Connecting Three Thermistors on a Single Panel

Connecting Two Terminal Panels for Reference Temperature Measurements

In this configuration, two Terminal Panels are used with an HP E1413 to provide up to 64 channels for temperature measurements. Make the following connections on the thermistor terminal blocks of both Terminal Panels:

- Connect HI-I to HI-TI of the First Terminal panel
- Connect LO-I to LO-TI of the First Terminal Panel
- Connect LO-TI of the First Terminal Panel to HI-TI of the Second Terminal Panel
- Connect LO-TI of the Second Terminal Panel to LO-I of the First Terminal Panel

This provides the excitation current to all six on-board thermistors on the Terminal Panels. Figure 3-7 shows the connection for two Terminal Panels.

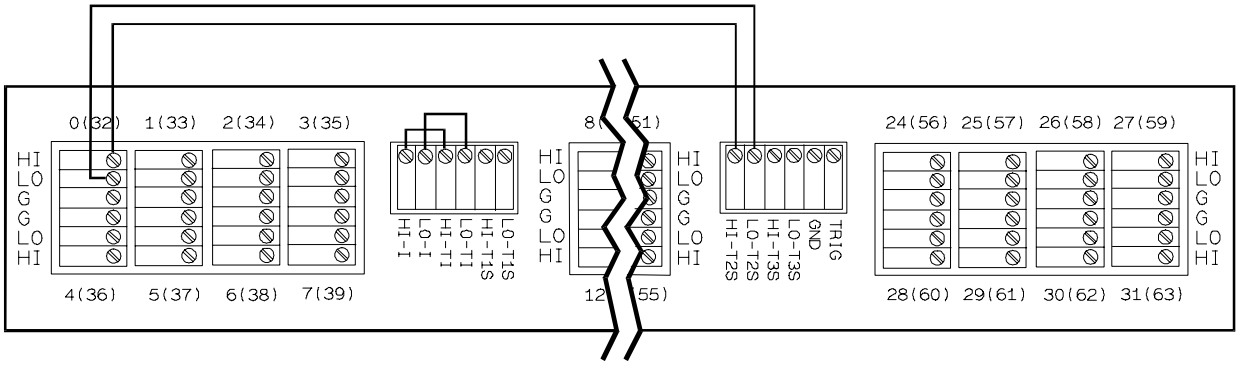


Two HP E1586 Terminal Panels
using All Thermistors on Each Panel

Figure 3-7. Connecting Six Thermistors on Two Panels

Measuring Using the Center Thermistor

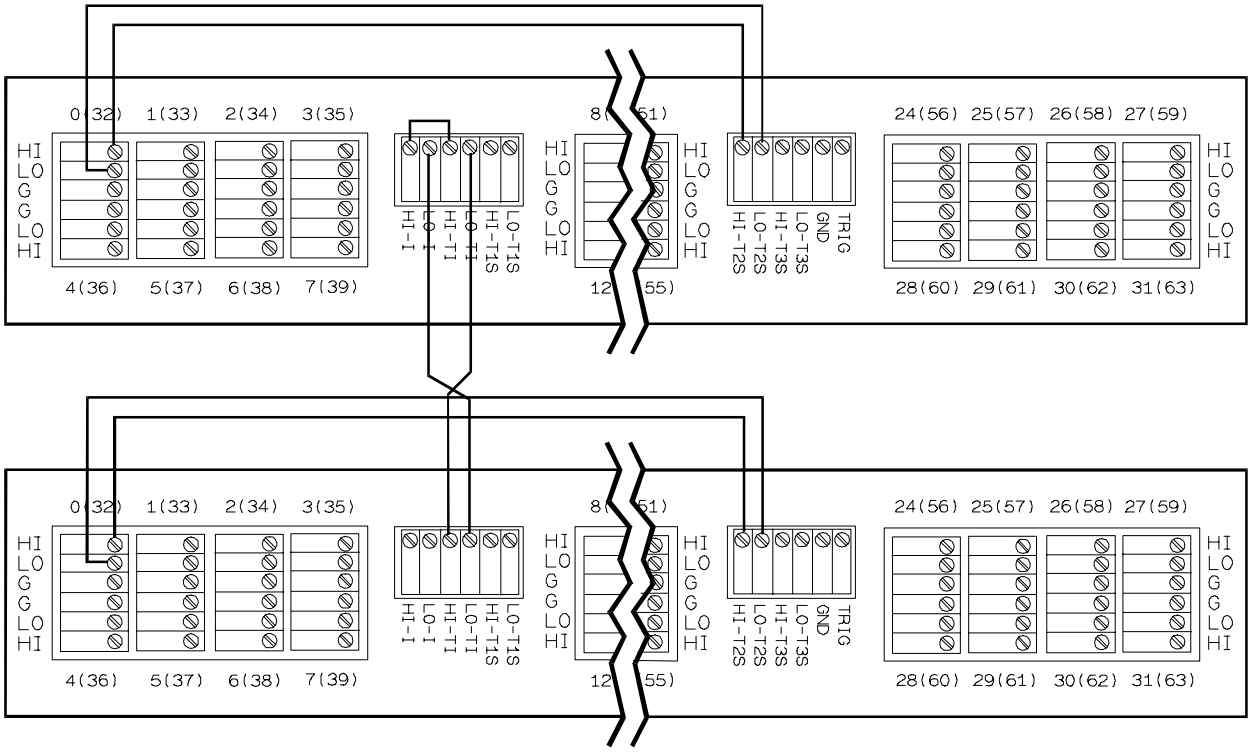
To measure the center thermistor, select an HP E1413C channel as a reference channel. Connect the reference channel's HI and LO to the center thermistor (thermistor 2) HI-T2S and LO-T2S terminals, respectively. These connections are shown in Figure 3-8.



One HP E1586 Terminal Panel
Measuring One Thermistor
on Reference Channel 100

Figure 3-8. Center Thermistor Measurements on a Single Panel

If two Terminal Panels are used, each panel must be connected as above so that both panels provide reference temperature measurements. These connections are shown in Figure 3-9.



Two HP E1586 Terminal Panels
Measuring One Thermistor on Each Panel
on Reference Channels 100, 132

Figure 3-9. Center Thermistor Measurements on Multiple Panels

The following example uses two Terminal Panels to measure type K thermocouples.

```
SENS:REF  THER,5000,1,(@100,132)
```

Measures reference temperature measurements on channels 100 and 132.

```
SENS:FUNC:TEMP  TC,K,.06,(@101:131,133:163)
```

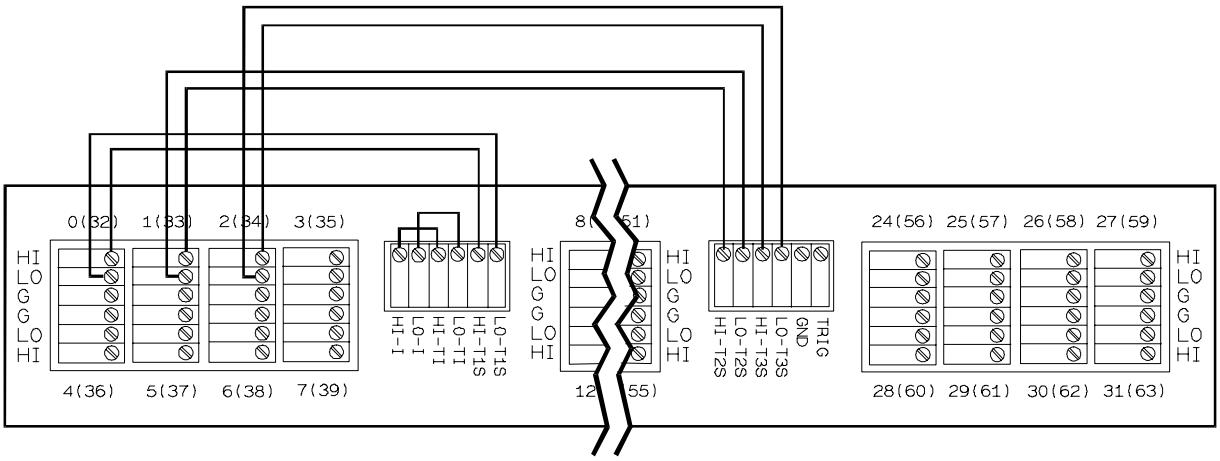
Defines channels for temperature measurements.

```
ROUT:SEQ:DEF  (@100,101:131,132,133:163)
```

Defines the scan list.

Measuring Using the Left, Center, and Right Thermistors

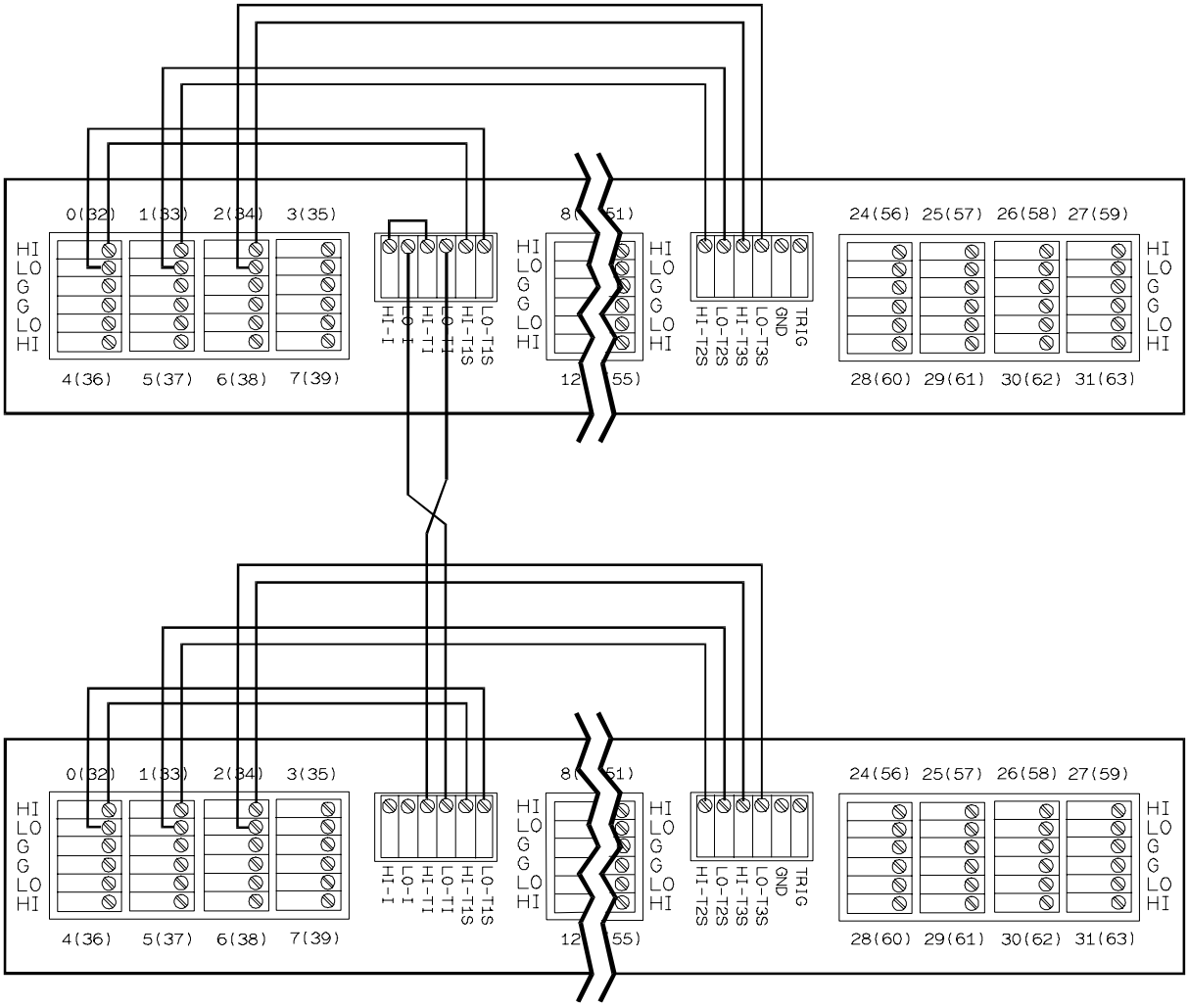
To measure all three thermistors, select three HP E1313/E1413 channels as reference channels. The recommended method is to use thermistor 1, the left thermistor (HI-T1S and LO-T1S), as a reference for channels 0 - 7 (or 32 - 39), thermistor 2, the center thermistor (HI-T2S and LO-T2S), for channels 8 - 23 (or 40 - 47), and thermistor 3, the right thermistor (HI-T3S and LO-T3S), for channels 24 - 31 (or 56 - 63). Connect the reference channel's HI and LO terminals to the appropriate thermistor terminals (e.g., channel 100 HI and LO terminals to thermistor 1 HI-T1S and LO-T1S terminals, respectively). These connections are shown in Figure 3-10.



One HP E1586 Terminal Panel
Measuring Three Thermistors
on Reference Channels 100, 101, 102

Figure 3-10. Left, Center, and Right Thermistor Measurements on a Single Panel

If two Terminal Panels are used, each panel must be connected as above so that both panels provide reference temperature measurements. These connections are shown in Figure 3-11.



Two HP E1586 Terminal Panels
Measuring One Thermistor on Each Panel
on Reference Channels 100, 101, 102
132, 133, 134

Figure 3-11. Left, Center, and Right Thermistor Measurements on Multiple Panels

The next two examples measure type K thermocouples. One example uses a single Terminal Panel and the other uses two Terminal Panels.

Example Using a Single Terminal Panel:

```
SENS:REF  THER,5000,1,(@100,101,102)
```
Measures reference temperature measurements on channels 100 to 102.

```
SENS:FUNC:TEMP  TC,K,.06,(@103:131)
```
Defines channels for temperature measurements.

```
ROUT:SEQ:DEF  (@100,103:107,101,108:123,102,124:131)
```
Defines the scan list.

Example Using Two Terminal Panels:

```
SENS:REF  THER,5000,1,(@100,101,102,132,133,134)
```
Measures reference temperature measurements on channels 100 to 102.

```
SENS:FUNC:TEMP  TC,K,.06,(@103:131,135:163)
```
Defines channels for temperature measurements.

```
ROUT:SEQ:DEF  (@100,103:107,101,108:123,102,
124:131,132,135:139,133,140:155,134,156:163)
```
Defines the scan list.

Note that each reference channel (100, 101, 102, 132, 133, 134) immediately precedes its associated measurement channels in the scan list.

Notes

Chapter 4
Understanding the HP E1313/E1413

Except where noted, all references to the HP E1413 apply to the HP E1313. This chapter introduces more advanced SCPI programming procedures for the HP A/D Scanning Converter and includes programming examples. Chapter contents include:

Advanced FIFO Data Retrieval

The discussion of FIFO data access in Chapter 3 ("Step 10. Retrieving Data") assumed that the number of readings to be transferred from the HP E1313/E1413 is a known quantity. The [SENSe:]DATA:FIFO:PART? command works well in that situation. This section is applicable when:

- You do not 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 cannot 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 (INITiate:CONTinuous is ON or TRIGger:COUNt is set to INFinite).

- Your system contains several HP E1313/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 [SENSe:]DATA:FIFO:PART? or [SENSe:]DATA:FIFO[:ALL]? commands.

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

Figure 4-1 shows program flow in the FIFO data retrieval section for a single HP E1313/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 (INITiate sets the Measuring bit checked in the first decision block in Figure 4-1).

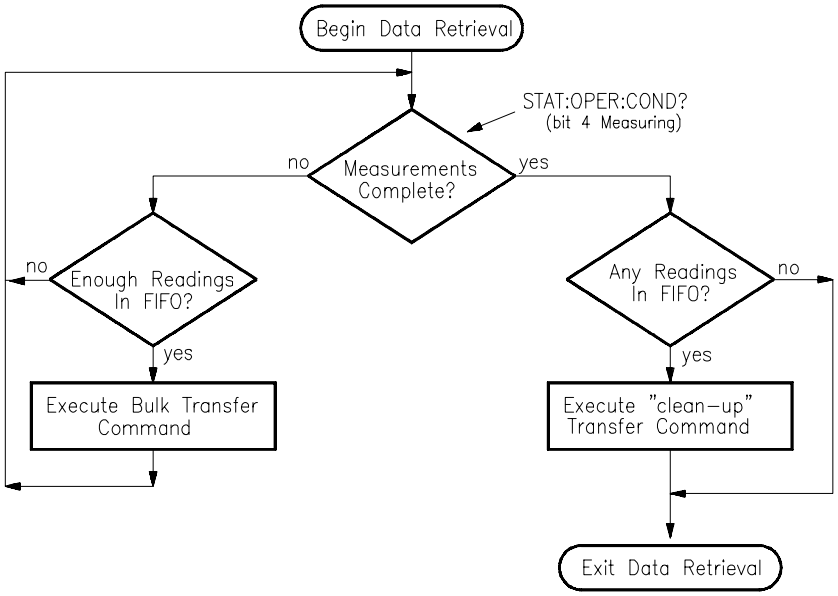


Figure 4-1. General Form of FIFO Data Retrieval

Comments

- 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 readings (count depends on which status FIFO command used), flow will fall through to execute the FIFO "bulk 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.

Choosing the Data Retrieval Method

There are two command sets available to retrieve large blocks of data as diagrammed in Figure 4-2.

Controlled Reading Count

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

SENSe:DATA:FIFO:COUNt?	*Determine the number of readings in the FIFO.*
SENSe:DATA:FIFO:PART? <*n_readings*>	*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 of the FIFO Data Retrieval 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.

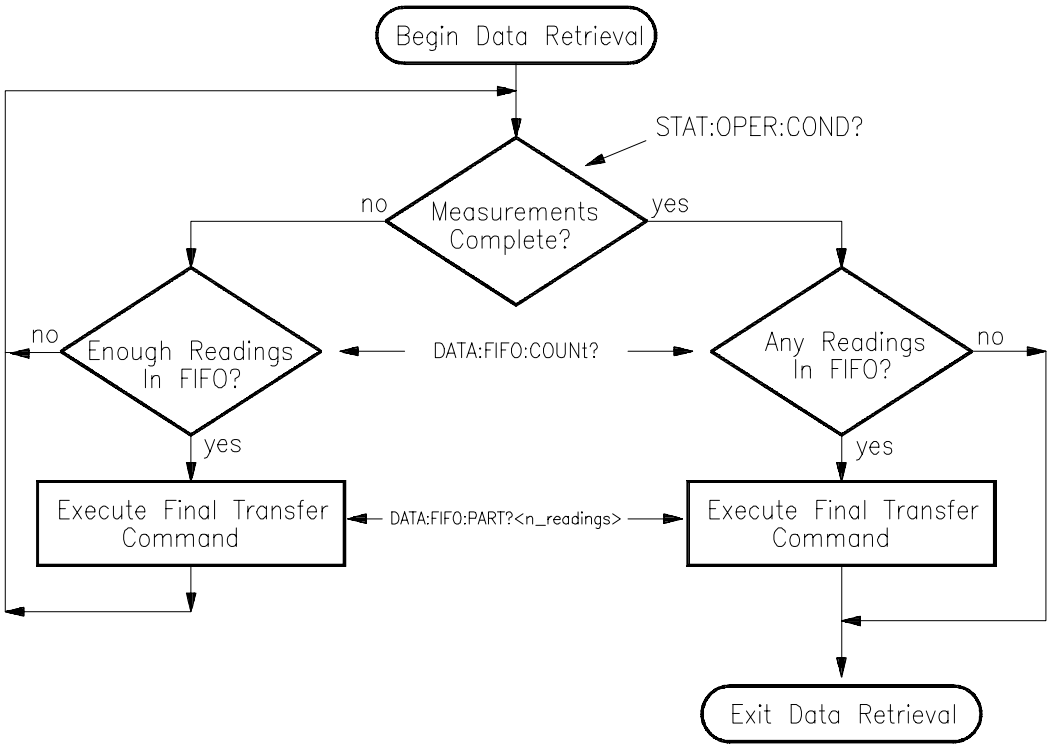


Figure 4-2. Controlling Reading Count

Example Command Sequence

```
*RST
SENS:FUNC:VOLT:DC  AUTO,(@100:163)   This is a *RST default.
ROUT:SEQ:DEF  LIST1,(@100:163)       This is a *RST default.
ROUT:SCAN  LIST1                     This is a *RST default.
SAMP:TIM  LIST1,.00001
ARM:SOUR  IMM
TRIG:COUN  16384
TRIG:TIM  .0007
TRIG:SOUR  TIM
FORM:DATA  REAL,32
INIT:IMM
    The following loop reads number of readings in FIFO while module is scanning.
loop while "measuring" bit is true   See STAT:OPER:COND bit 4.
  SENSe:DATA:FIFO:COUNt? <n_readings>n_readings is an unsigned 16-bit
                                     integer.
  if <n_readings> = 16384
      SENSe:DATA:FIFO:PART? <read_data>Read_data is a 32-bit floating
                                     point array.
  end if
end loop
    The following checks for readings remaining in FIFO after "measuring" false.
SENSe:DATA:FIFO:COUNt? <n_readings>
if n_readings                        If any readings...
  SENSe:DATA:FIFO:PART? <read_data>  Get readings from FIFO.
end if
```

Fastest Reading Transfer

This command set is made up of the following commands:

SENSe:DATA:FIFO:COUNt:HALF?	*Determine if there are at least 32,768 readings in the FIFO.*
SENSe:DATA:FIFO:HALF?	*Transfer one half of the FIFO (32,768 readings).*
SENSe:DATA:FIFO:ALL?	*Return the remaining readings from the FIFO **after** scanning is complete.*

What makes this method fastest is that [SENSe:]DATA:FIFO:HALF? and [SENSe:]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 [SENSe:]DATA:FIFO[:ALL]? will complete even if there are no readings in the FIFO, you do not need 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.

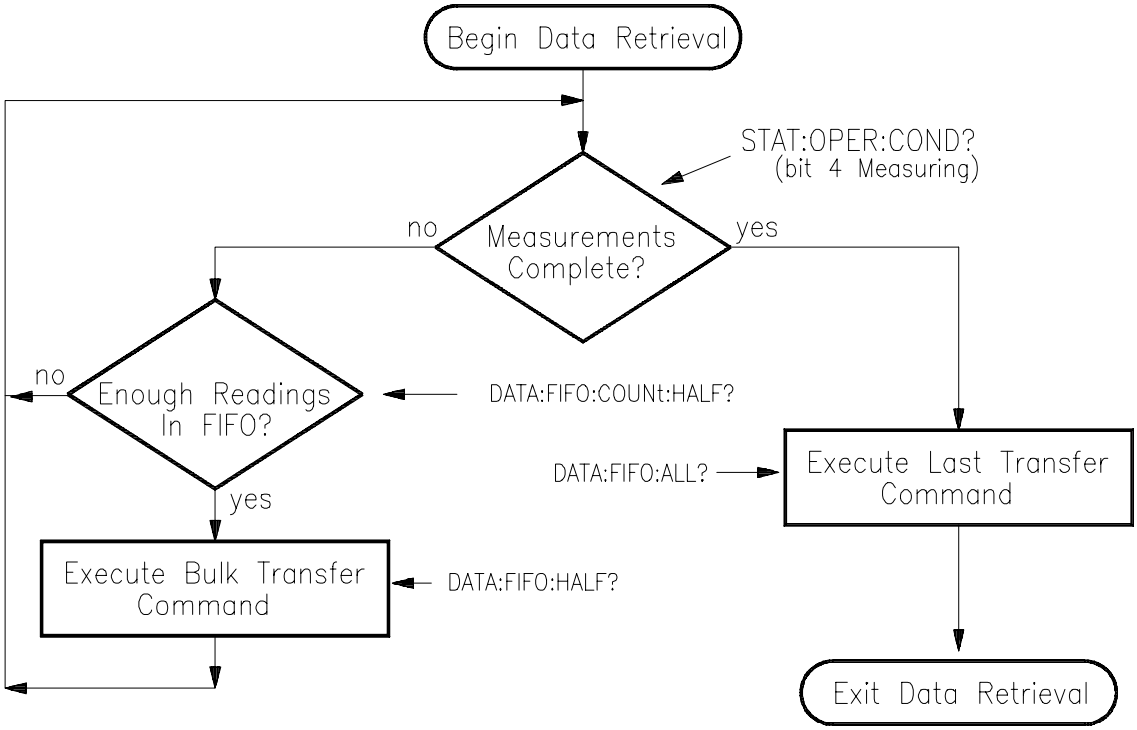


Figure 4-3. Fastest Reading Transfer

Example Command Sequence

```
*RST
SENS:FUNC:VOLT:DC  AUTO,(@100:163)   This is a *RST default.
ROUT:SEQ:DEF  LIST1,(@100:163)       This is a *RST default.
ROUT:SCAN  LIST1                     This is a *RST default.
SAMP:TIM  LIST1,.00001
ARM:SOUR  IMM
TRIG:COUN  16384
TRIG:TIM  .001
TRIG:SOUR  TIM
FORM:DATA  REAL,32
INIT:IMM
    The 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.
  SENSe:DATA:FIFO:COUNt:HALF?  <half_flag>half_flag is a 16-bit integer.
  if half_flag
      SENSe:DATA:FIFO:HALF?  <read_data>read_data is a 32-bit floating
                                        point array.
  end if
end loop
    The following reads remaining data, if any, after scanning stops.
SENSe:DATA:FIFO:ALL?  <read_data>
```

Controlling Data Conversion and Destination

For ROUTe:SEQuence:DEFine, 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 E1313/E1413, card number becomes the channel data modifier. The value can range from 1 through 7. The value controls whether Engineering Unit (EU) conversion is performed and the internal destination of the resulting value. Table 4-1 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 (CVT).
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 do not 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:

```
ROUT: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 (three if Sample and Hold SCP installed) 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 (TRIGger:SOURce 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 ROUTe:SCAN <*scan_list*>) becomes effective after the INITiate 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 TRIGger:COUNt specifies. Note that TRIGger: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 only set at the start of the counted scan. This means that the current scan list can not be changed while the module is scanning. However, by specifying LISTL as the current scan list before INITiate:CONTinuous ON, the sequence of scan lists specified in LISTL will be executed.

- The "Continuous Mode" is set when TRIGger:SOURce is set to IMMediate, and INITiate:CONTinuous is set to ON. Note that executing INITiate:CONTinuous ON not only sets the Continuous Mode but also initiates the module. To stop at the end of the current scan, execute the INITiate:CONTinuous 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 cannot be changed "on-the-fly". However, by specifying LISTL as the current scan list before INITiate:CONTinuous ON, the sequence of scan lists specified in LISTL will be executed. This mode is the fastest since the module does not need to read the current scan list or check for the occurrence of a trigger event.

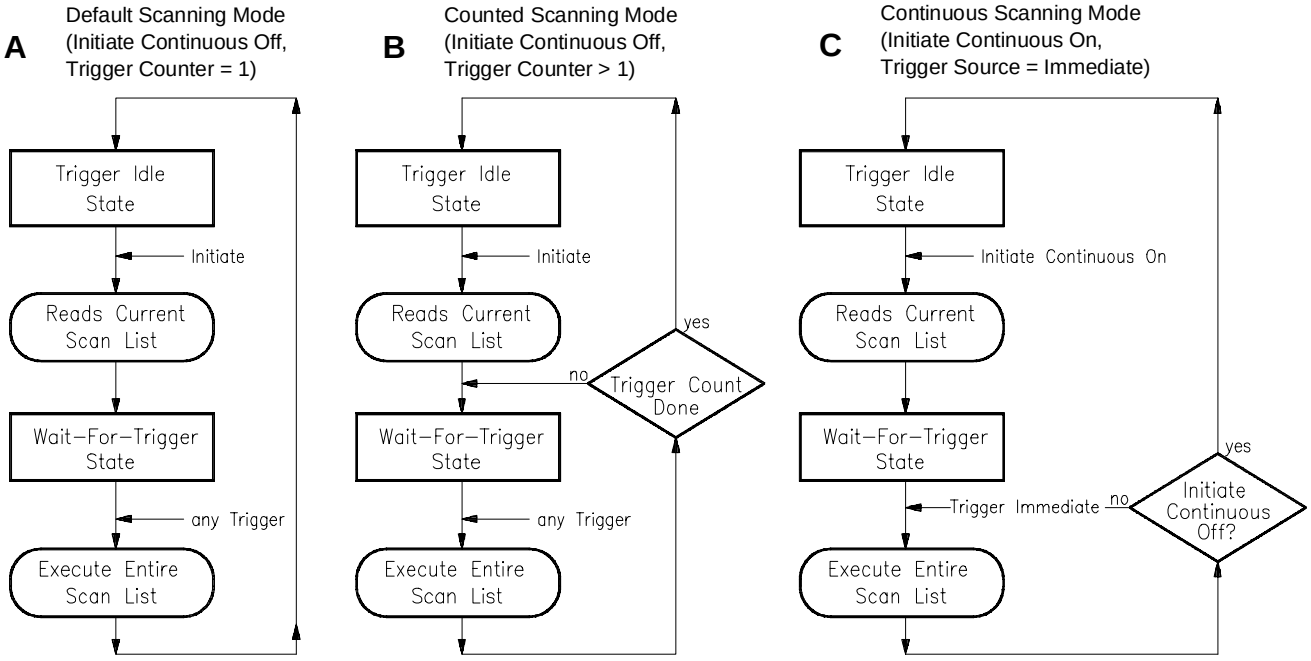


Figure 4-4. Scanning Modes

See the ROUTe:SCAN command and Figure 5-3 as well as the TRIGger:TIMer:MODE command and Figures 5-7 and 5-8 in Chapter 5 for full details.

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

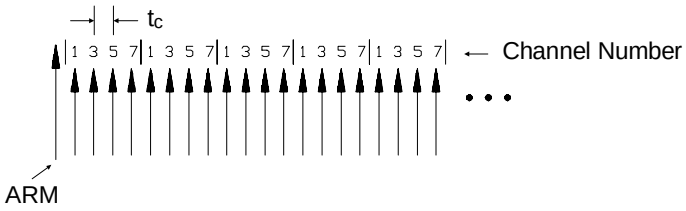
- Use program *verif.cs* for the default mode.
- Use program *counted.cs* for the counted mode.
- Use program *through.cs* for the continuous mode.

Triggering and Scanning Modes

The following section shows the various ways the HP E1313/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 HP E1313/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 IMM

ARM:SOUR <*source*> *<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*

INIT:CONT ON

Starting

ARM event, either hardware or software
Acquisition starts

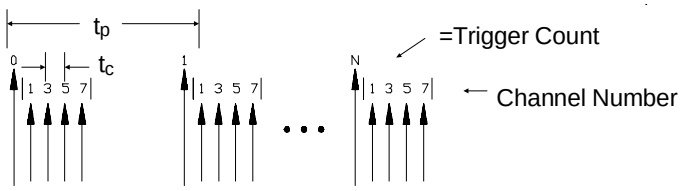
Stopping

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 .1 ms 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 10 µs to 32 ms.

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 - TRIGger sequence is executed. Trigger 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 HP E1313/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*>	*<n> can be 1 - 4.*
SAMP:TIM LIST<*n*>,<t_c>	*<t_c> can be 10µs to 32.768ms.*
TRIG:SOUR TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
INIT:IMM	

Starting

ARM event, either hardware or software

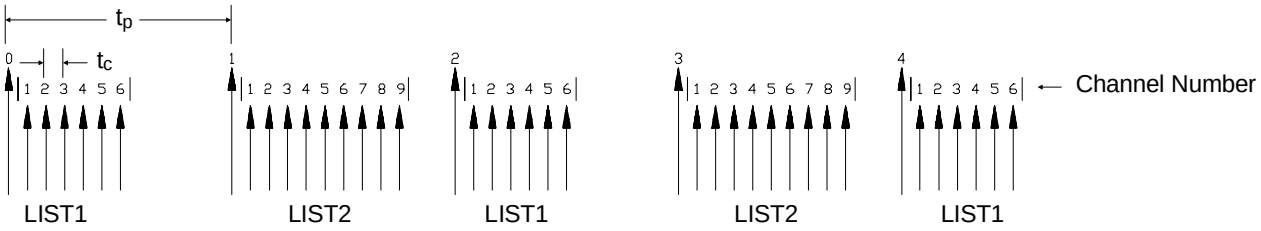
Acquisition starts

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 (must use ABOR if TRIG:COUNt is INF).*

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 1,024 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 HP E1313/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>	*Set sample time for lists in LISTL.*
TRIG:SOUR TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
INIT:IMM	

Starting

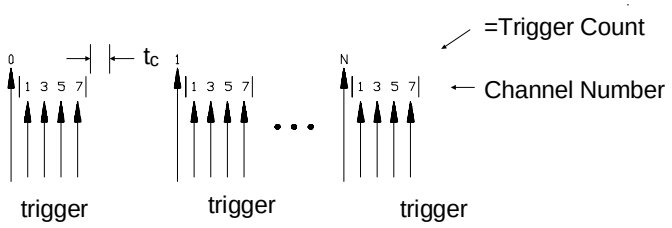
ARM event, either hardware or software
Acquisition starts

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 (must use ABOR if TRIG:COUNt is INF).*

Externally Paced Scans

This mode is much like the Timer Paced Scans except the trigger source is external to the HP E1313/E1413. The trigger could come from a software command such as TRIGger, 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 HP E1313/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*>	*<n> can be 1 - 4.*
SAMP:TIM LIST<*n*>,<t_c>	*<t_c> can be 10µs to 32.768ms.*
TRIG:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, or TTLTrg0 - TTLTrg7.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
INIT:IMM	

Starting

Trigger event, either hardware or software

Acquisition starts

Stopping Before Trigger Count is Reached

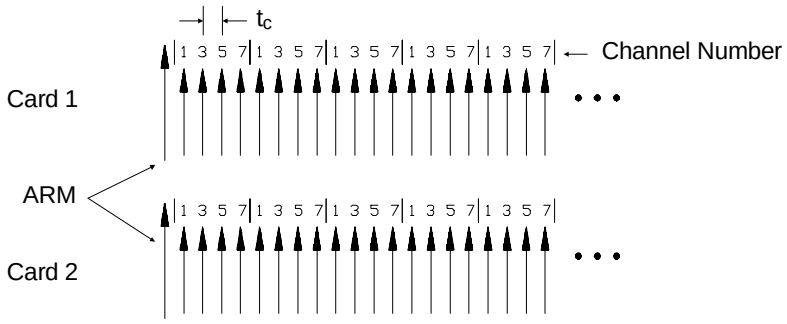
Remove Trigger Source	*Stops at end of scan list, still initiated.*
or	
TRIG:SOUR HOLD	*Stops at end of scan list, still initiated.*
TRIG:IMM	*One more scan, now idle.*
or	
ABOR	*Stops immediately (must use ABOR if TRIG:COUNt is INF).*

Synchronizing Multiple Cards

In this section we show you how to synchronize two or more HP E1313/ 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 (TRIGger:SOURce IMMediate and INITiate:CONTinuous 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 HP E1313/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*>,<t_c> *<t_c> can be 10µs to 32.768ms.*

TRIG:SOUR IMM

ARM:SOUR <*source*> *<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*

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 IMM

ARM:SOUR <*source*> *<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*

INIT:CONT ON

Starting

ARM event, either hardware or software

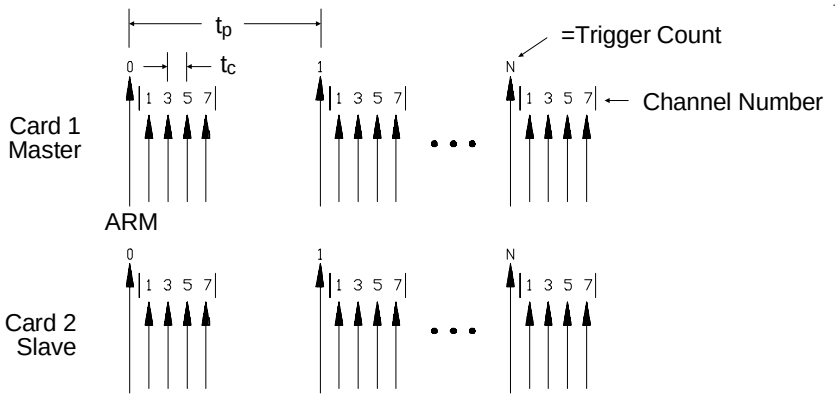
Acquisition starts

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 HP E1313/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 TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
OUTP:TTLT:SOUR TRIG	*Will drive TTLT line if triggered.*
OUTP:TTLT<*n*>:STAT ON	*Enable to drive selected ttltrg line. <n> can be 1 through 7.*
INIT:IMM	

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<*n*>	*Selects TTLTrg trigger source. <n> can be 1 through 7.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
INIT:IMM	

Starting

ARM event, either hardware or software

Acquisition starts

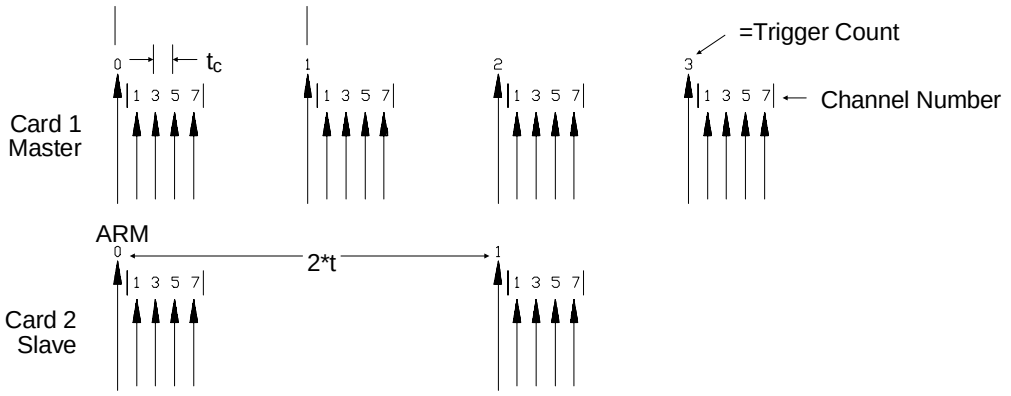
Stopping Before Trigger Count is Reached

TRIG:SOUR HOLD	*Stops at end of scan list, still initiated.*
TRIG:IMM	*One more scan, now idle.*
or	
ABOR	*Stops immediately (must use ABOR if TRIG:COUNt is INF).*

Timer Based Scans at Different Rates

This example shows how to use the HP E1313/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 $\frac{1}{TRIGger:COUNt}$

Also shown is the TRIGger:TIMer:MODE command. When TRIGger:TIMer:MODE is set to SYNC, the trigger timer keeps running even after the trigger count is reached. When TRIGger:TIMer: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 HP E1313/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 TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:TIM:MODE SYNC	*Keep timer running during INIT:CONT ON.*
TRIG:COUN 2	
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
OUTP:TTLT<*n*>:SOUR FTR	*1st trig will source TTLTrg.*
OUTP:TTLT<*n*>:STAT ON	*Enable to drive selected TTLTrg line. <n> can be 1 through 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*>,<t_c>	*<t_c> can be 10µs to 32.768ms.*
TRIG:SOUR TTLTrg<*n*>	*Selects TTLTrg trigger source.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
INIT:IMM	

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	
ABOR	*Stops immediately (must use ABOR if TRIG:COUNt is INF).*

Using Automatic Scan List Sequencing (List-of-Lists)

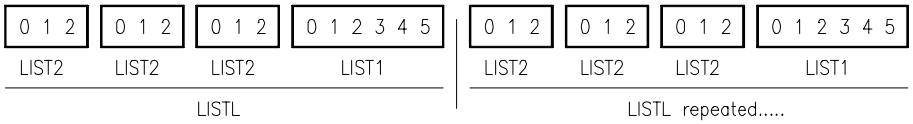
It is not unusual in data acquisition that you will want to scan groups of channels at different rates. Some of your system's channels may measure fast changing signals that need to monitored at high rates while other channels measure slowly changing signals that can be scanned less often.

The HP E1313/E1413 provides an easy way to scan groups of channels at up to four different rates. The module not only has four scan lists (LIST1 through LIST4) in which you define lists of channels, it also has LISTL which you define as a list of scan lists (therefore the term List-of-Lists). When LISTL is selected as the current "scan list", a trigger will cause the module to execute each scan list defined in LISTL in the order it was defined. The overall procedure is:

1. Define each scan list to be executed using the command ROUTe:SEQuence:DEFine LIST*n* (@<*ch_list*>).
2. Define LISTL with the sequence of scan lists to execute using the command ROUTe:SEQuence:DEFine LISTL (@<*ch_list*>).
3. Set LISTL as the current scan list using the command ROUTe:SCAN LISTL.
4. Set up the trigger system for continuous mode or counted mode (see previous section).
5. Trigger the module to start scanning.

A Simple Example

What controls the relative rates that the channel groups are scanned is the frequency that individual channels appear in each scan list and the frequency that these scan lists appear in LISTL. As a very simple example let's say you want to scan channels 0 through 3 four times as often as channels 4 through 7. Since there are two groups we will use two scan lists. The scan list to be executed least often contains the channels from both the high rate group and the low rate group [ROUTe:SEQuence:DEFine LIST1, (@100:107)]. The scan list to be executed most often contains only the channels from the high rate group [ROUTe:SEQuence:DEFine LIST2, (@100:103)]. Now we will define LISTL to scan the high rate group four times for every scan of the low rate group. Define LISTL to reference scan list 2 three times and scan list 1 once [ROUTe:SEQuence:DEFine LISTL,(@2,2,2,1)]. When triggered, the module will scan these channels:



Programming Four Different Rates

To program four groups of channels to scan at different rates we will use all four scan lists. Our example:

- scan channels 0 through 15, 300 times each second
- scan channels 16 through 31, 150 times each second
- scan channels 32 through 47, 50 times each second
- scan channels 48 through 63, 10 times each second

We will define the minimum rate group, channels 0-63 in scan list 1. Scan list 2 will be defined as channels 0-47. Scan list 3 will be channels 0-31. Scan list 4 will be only channels 0-15. As you can see, the higher rate scan lists are subsets of the lowest rate scan list.

The maximum rate is 30 times the minimum rate so we will need thirty entries in LISTL. LISTL will reference the maximum rate group of channels 30 times and the min rate group only once. The group that is to be scanned 150 times per second will be referenced 15 times in LISTL (every other scan). The group that is to be scanned 50 times each second will be referenced 5 times (every sixth scan).

```
ROUT:SEQ:DEF  LIST1,(@100:163)        Minimum rate group (*RST default).
ROUT:SEQ:DEF  LIST2,(@100:147)        150 Hz group.
ROUT:SEQ:DEF  LIST3,(@100:131)        50 Hz group.
ROUT:SEQ:DEF  LIST4,(@100:115)        Maximum rate group.
ROUT:SEQ:DEF  LISTL,(@4,3,4,3,4,2,4,3,4,3,4,2,4,3,4,3,4,2,4,3,4,3,4,2,4,3,4,3,4,1)
```

Setting the Absolute Scan Rate

So far we have only set relative rates that channels are measured. To set an absolute channel scanning rate, we must control the interval between execution of scan lists. To do this we use the trigger timer, so we select the trigger timer as the trigger source:

```
TRIG:SOUR  TIM                        Trigger source is timer.
```

We then set the trigger interval using the TRIGger:TIMer[:PERiod] <*trig_interval*> command. In our example we said that we want the highest rate channels to be scanned 300 times each second. The trigger interval would then need to be 3.3333 msec:

```
TRIG:TIM  3.3333ms                    Scan list executed at 300 Hz.
```

To minimize time skew between the first and last channel in each scan list, we will set the sample timer period as small as possible (10µs):

```
SAMP:TIM  LISTL,10us                  Channel-to-channel pacing 10µsec.
```

Using the Status System

The HP E1313/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 that the three Status Groups (Operation Status, Questionable Data, and the Standard Event Groups) and the Output Queue all send summary information to the Status byte. By this method the Status Byte can report many more events than its eight bits would otherwise allow. Figure 4-6 shows the Status system in detail.

Note The C-SCPI program *status.cs* on your C-SCPI driver tape, the C and Visual BASIC programs *status.xx* on the driver/example program disk show how to use the status system.

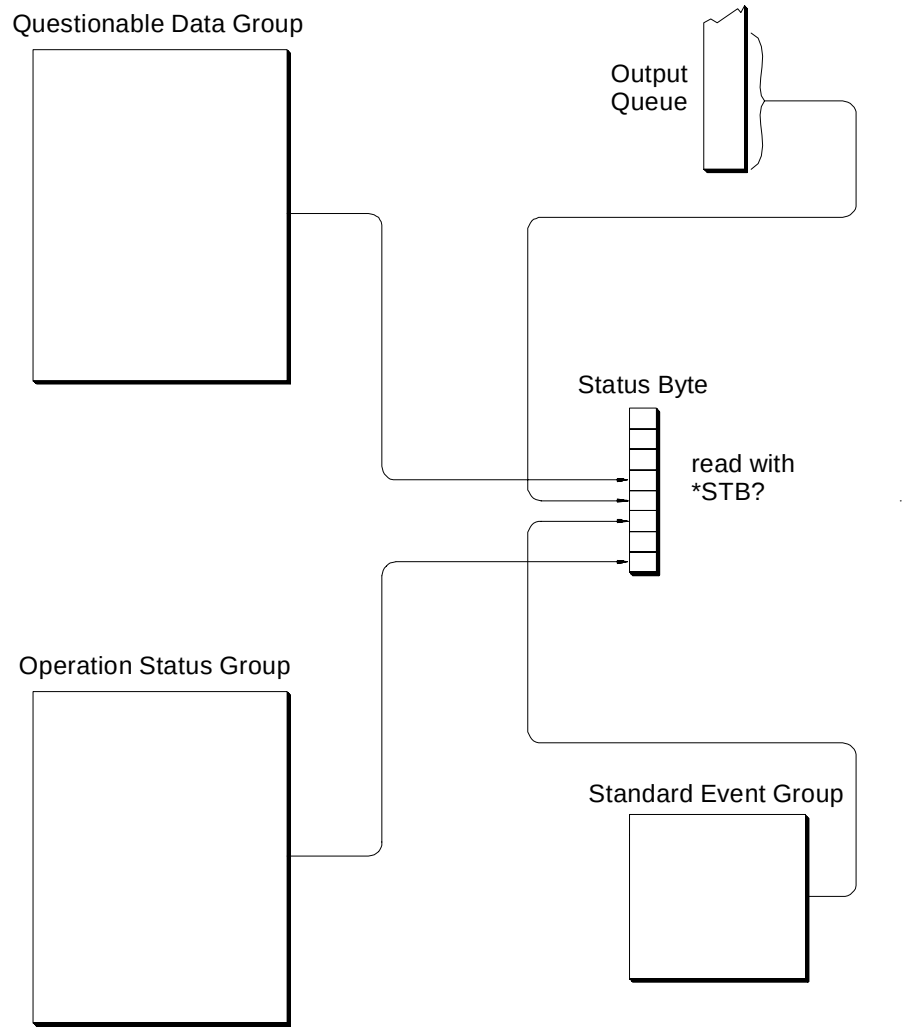


Figure 4-5. Simplified Status System Diagram

Enabling Events to be Reported in the Status Byte

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

Configuring the Transition Filters

Figure 4-6 shows that the Condition Register outputs are routed to the input of the Negative Transition and Positive Transition Filter Registers. For space reasons they are shown together but are controlled by individual SCPI commands. Table 4-5 is the truth table for the Transition Filter Registers.

Table 4-5. Transition Filter Registers

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

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

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.

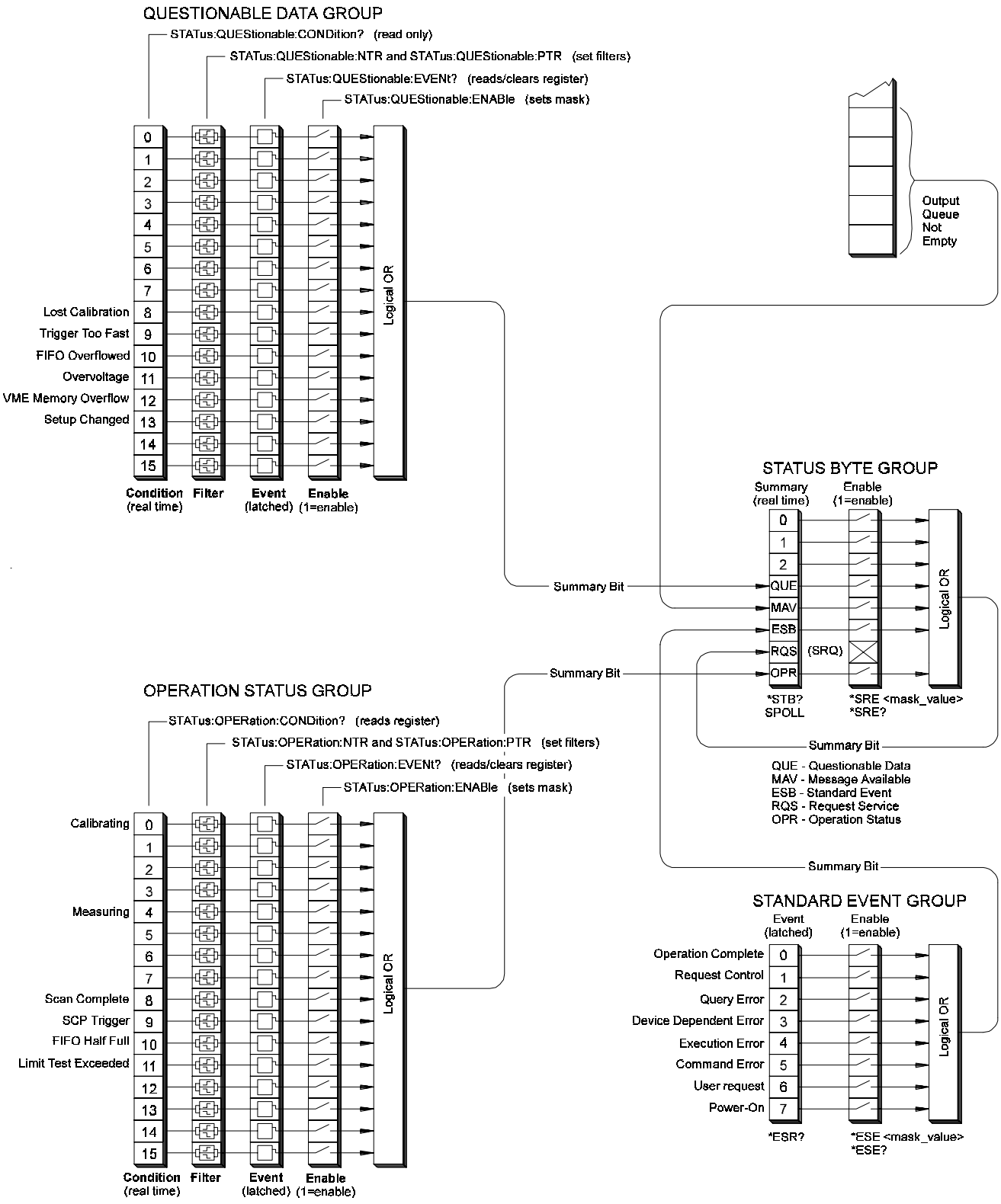


Figure 4-6. HP E1313/E1413 Status System

Table 4-2. Questionable Data Group

Bit	Bit Value	Event Name	Description
8	256	Lost Calibration	At *RST or Power-on Control Processor has found a checksum error in the Calibration Constants. Read error(s) with SYST:ERR? command and re-calibrate areas that lost constants.
9	512	Trigger Too Fast	Scan not complete when another trigger event received.
10	1024	FIFO Overflowed	Attempt to store more than 65,024 readings in FIFO.
11	2048	Overvoltage (Detected on Input)	If the input protection jumper has not been cut, the input relays have been opened and *RST is required to reset the module. Overvoltage will also generate an error.
12	4096	VME Memory Overflow	The number of 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).

Table 4-3. Operation Status Group

Bit	Bit Value	Event Name	Description
0	1	Calibrating	Set by CAL:TARE, and CAL:SETup. Cleared by CAL:TARE?, and CAL:SETup?. Set while *CAL? executing, then cleared.
4	16	Measuring	Set when instrument INITiated. Cleared when instrument returns to Trigger Idle State.
8	256	Scan Complete	Set when each pass through a Scan List is completed (may not indicate that all readings have been taken when INIT:CONT is ON).
9	512	SCP Trigger	Reserved for future HP E1413 SCPs.
10	1024	FIFO Half Full	FIFO contains at least 32,768 readings.
11	2048	Limit Test Exceeded	One or more limit tests exceeded.

Table 4-4. Standard Event Group

Bit	Bit Value	Event Name	Description
0	1	Operation Complete	*OPC command executed and instrument has completed all pending operations.
1	2	Request Control	Not used by HP E1313/E1413.
2	4	Query Error	Attempting to read empty output queue or output data lost.
3	8	Device Dependent Error	A device dependent error occurred. See Appendix B.
4	16	Execution Error	Parameter out of range, or instrument cannot execute a proper command because it would conflict with another instrument setting.
5	32	Command Error	Unrecognized command or improper parameter count or type.
6	64	User Request	Not used by HP E1313/E1413.
7	128	Power-On	Power has been applied to the instrument.

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.

Operation Group Example

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

```
STAT:OPER:PTR  32766
```
All ones in Positive Transition Filter Register except bit 0 = 0.

```
STAT:OPER:NTR  1
```
All zeros in Negative Transition Filter Register except bit 0 = 1.

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

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 STATus:QUEStionable:EVENt? command. This returns 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 SYSTem:ERRor? 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 SYSTem:ERRor? command to read these.
Bit 7 (OPR) bit value 128_{10}	Read the Operation Status Group's Event Register using the STATus:OPERation: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:PRES	*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 4-6 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:EVEN?	*Questionable Data Group Event Register.*
STAT:OPER:EVEN?	*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.*

Updating the Status System and VXIbus Interrupts

The driver needs to update the Status systems information whenever the status of the HP E1313/E1413 changes. This update is always done when the Status system is accessed, or when CALibrate:…, INITiate:…, or ABORt commands are executed. Most of the bits in the Operation and Questionable Registers represent conditions which can change while the HP E1313/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 INITiate:… 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 E1313/E1413, the HP E1313/E1413 interrupts can be used.

The HP E1313/E1413 can send VXIbus 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 E1313/E1413.
- Scan complete. The HP E1313/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 INITiate:CONTinuous. This clears the Measuring bit in the Operation Register.

These HP E1313/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 E1313/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:OPERation:ENABle, STATus:QUEStionable:ENABle, and STATus:PRESet commands.

Each of the interrupting conditions listed previously has a corresponding bit in the Questionable or Operation Registers. If that bit is enabled via the STATus:OPERation:ENABle or STATus:QUEStionable:ENABle command to be a part of the group summary bit, it will also enable the HP E1313/E1413 interrupt for that condition. If that bit is not enabled, the corresponding interrupt will be disabled.

Sending the STATus:PRESet command will disable all the interrupts from the HP E1313/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:OPERation:ENABle or STATus:QUEStionable: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, STATus:OPERation:ENABle, or STATus:QUEStionable:ENABle 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 *C-SCPI Users Guide* for more details on the overlapped mode and using interrupts in the C-SCPI environment.

HP E1313/E1413 Background Operation

The HP E1313/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 INITiate[:IMMediate] or INITiate:CONTinuous ON commands are given, the HP E1313/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 E1313/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, that is, the scan list can be changed using ROUTe:SEQuence:DEFine. The INITiate:CONTinuous 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 E1313/E1413 is actively measuring.

Averaging Readings

The HP E1313/E1413 provides on-board averaging of readings. When you set averaging ON (CALCulate:AVERage[:STATe] ON), a reading returned from the FIFO and/or CVT represents the average of two or more measurements on that channel. Enabling averaging enables averaging for all channels. The number of measurements per reading is set using CALCulate:AVERage:COUNt 2 | 4 | 8 | 16 | 32 | 64 | 256.

Special Considerations

The following restrictions apply when performing channel averaging.

- Because each channel can be allocated only one memory location for intermediate calculation results, an individual channel number must appear in a scan list only once, and use of LISTL (List-of-Lists) is not allowed.
- In order to maintain high speed while averaging when CALCulate:AVERage[:STATe] is ON, channels must use manual ranging.

Example Command Sequence

ROUT:SEQ:DEF LIST1,(@100,101,102,103)	*Define scan LIST1.*
ROUT:SCAN LIST1	*Select scan LIST1.*
SAMP:TIM LIST1,<t_c>	*<t_c> can be 10µs to 32.768ms.*
TRIG:SOUR TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF but it MUST be ≥ CALC:AVER:COUNt.*
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
CALC:AVER:COUN 4	*Each reading averages of 4 measurements.*
CALC:AVER ON	*Enable averaging.*
INIT:IMM	

Starting

ARM event, either hardware or software
Acquisition starts

Stopping

Stops when Trigger Count is reached unless count is INF, then use ABORt

---

Note TRIGger:COUNt must be greater than or equal to CALCulate:AVERage:COUNt in order to complete the average calculation.

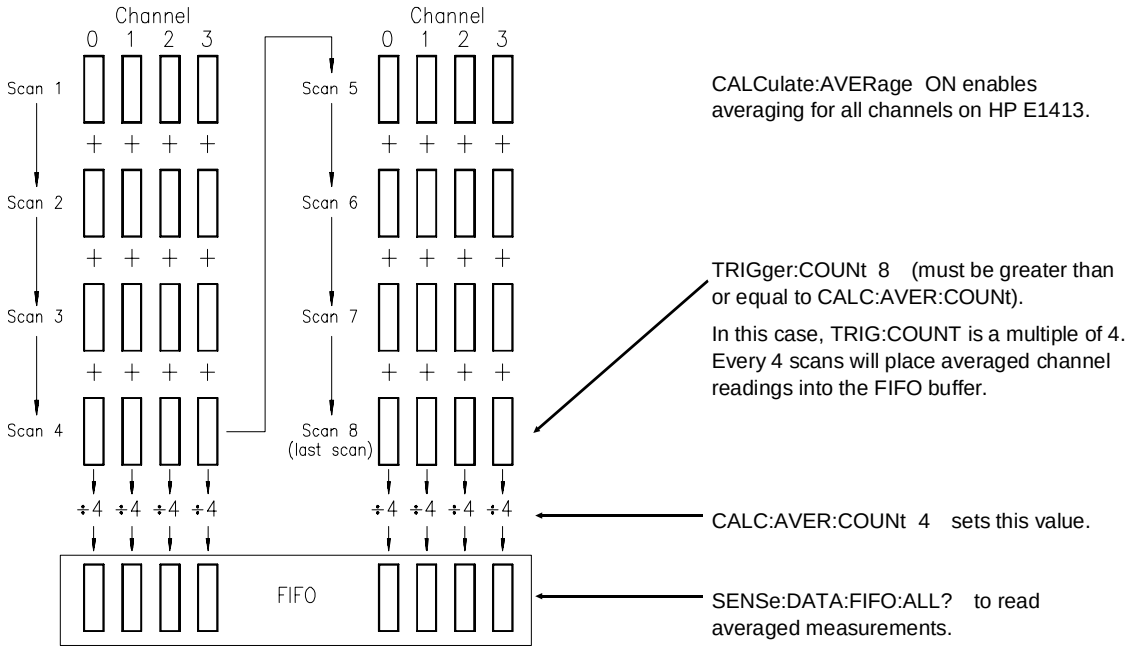


Figure 4-7. Averaging Channel Measurements

Limit Testing

The HP E1313/E1413 can test readings to see if they fall within test limits. You can program upper and lower test limits for each channel you wish to test. You can enable limits testing against upper and lower, upper limits only, or lower limits only.

Checking Results

You can query the results of limits testing several ways:

Checking Results with the CALCulate Subsystem

Composite	Returns a single value. A value of one indicates that one or more of the module's tested channels has exceeds its limits.
Channels	Returns an array of 64 values. Each value corresponds to one of the module's 64 channels. A value of one indicates that the channel has exceeded its limits.
Individual Channel	Returns a single value for a specific channel. A value of one indicates that the channel has exceeded its test limits.

There are two versions of each of the query types explained above:

Cumulative	Cumulative limit results are accumulated from the time the module is INITiated. If a channel has exceeded its limits since INITiate, even if it currently is within limits, the event is recorded.
Current	Current limit results indicate the status of limits tested during the last completed scan list.

Checking Results with the Status System

Polling	Poll by accessing bit 11 of the Operation Status Event Register. When bit 11 is set to one, one or more limits were exceeded.
Interrupt	If bit 11 of the Operation Status Enable Register is set, the Operation Summary bit will be sent to the Status Byte Register. An out-of-limits condition can then be reported by interrupt.

Example Command Sequence

ROUT:SEQ:DEF LIST<*n*>,(@101,103,105,107)	*<n> can be 1 - 4.*
SENS:FUNC:TEMP RTD,100,(@101,103)	*100Ω RTD temperature channels 1 and 3.*
SENS:FUNC:VOLT:DC (@105,107)	*Autorange voltage channels 5 and 7.*
ROUT:SCAN LIST1	*Default is autorange voltage.*
CALC:LIM:UPP:DATA 750,(@101,103)	*Upper limit set to 750 °C.*
CALC:LIM:LOW:DATA 450,(@101,103)	*Lower limit set to 450 °C.*
CALC:LIM:UPP:DATA .2,(@105,107)	*Upper limit set to +.2VDC.*
CALC:LIM:LOW:DATA -.01,(@105,107)	*Lower limit set to -.01VDC.*
CALC:LIM:STAT ON,(@101,103,105,107)	*Enable overall limit testing.*
CALC:LIM:UPP:STAT ON,(@101,103,105,107)	*Enable upper limit testing.*
CALC:LIM:LOW:STAT ON,(@101,103,105,107)	*Enable lower limit testing.*
SAMP:TIM LIST1,<t_c>	*<t_c> can be 10µs to 32.768ms.*
TRIG:SOUR TIM	
TRIG:TIM:PER <t_p>	*<t_p> can be .1ms - 6.5536 s.*
TRIG:COUN <*count*>	*<count> can be 1 - 32768 or INF.*
ARM:SOUR <*source*>	*<source> can be BUS, EXT, HOLD, IMM, or TTLTrg0 - TTLTrg7.*
INIT:IMM	

Starting

ARM event, either hardware or software
Acquisition starts

Stopping

Stops when Trigger Count is reached unless count is INF then use ABORt.

Querying Limit Test Results:

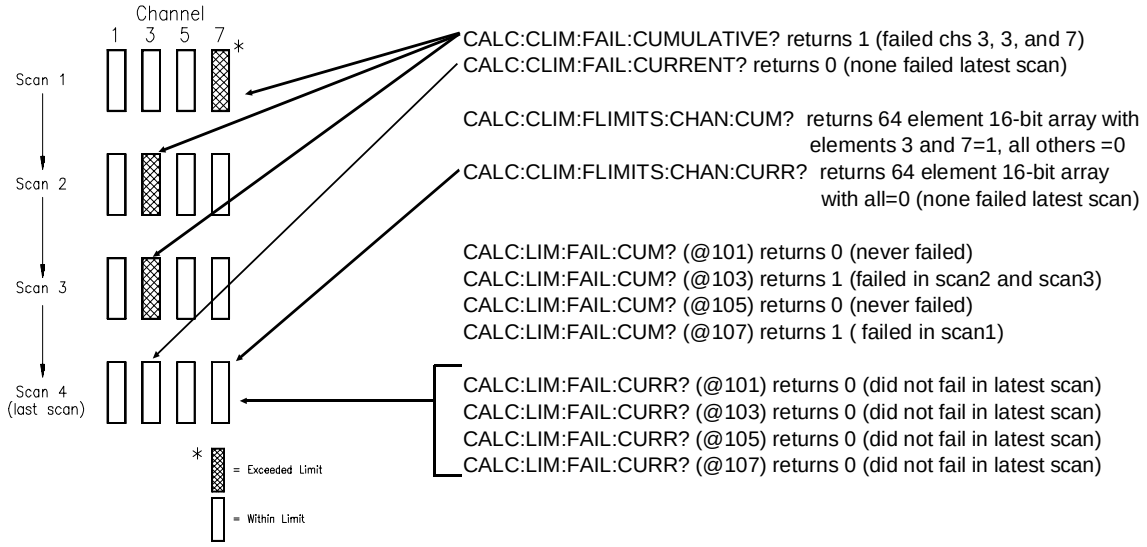


Figure 4-8. Querying Limit Test Results

Custom EU Conversion Tables

The HP E1313/E1413 provides for loading custom EU conversion tables. This allows you to have on-board conversion of transducers not otherwise supported by the HP E1313/E1413.

Standard EU Operation

The EU conversion tables built into the HP E1313/E1413 are stored in a "library" in the module's non-volatile Flash Memory. When you link a specific channel to a standard EU conversion using the [SENSe:]FUNCtion:… command, the module copies that table from the library to a segment of RAM allocated to the specified channel. When a single EU conversion is specified for multiple channels, multiple copies of that conversion table are put in RAM, one copy into each channel's Table RAM Segment. The conversion table-per-channel arrangement allows higher speed scanning since the table is already loaded and ready to use when the channel is scanned.

Custom EU Operation

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

Note

The *RST command clears all channel Table RAM segments. Custom EU conversion tables must be reloaded using the DIAGnostic:CUSTom:… commands.

Custom EU Tables

The HP E1313/E1413 uses two types of EU conversion tables, linear and piecewise. The linear table describes the transducer's response slope and offset ($y = mx + b$). The piecewise conversion table gets its name because it is actually an approximation of the transducer's response curve in the form of 512 linear segments whose end-points fall on the curve. Data points that fall between the end-points are linearly interpolated. The built-in EU conversions for thermistors, thermocouples, and RTDs use this type of table.

Custom Thermocouple EU Conversions

The HP E1313/E1413 can measure temperature using custom characterized thermocouple wire of types E, J, K, N, R, S, and T. The custom EU table generated for the individual batch of thermocouple wire is loaded to the appropriate channels using the DIAGnostic:CUSTom:PIECewise command. Since thermocouple EU conversion requires a "reference junction compensation" of the raw thermocouple voltage, the custom EU table is

linked to the channel(s) using the command [SENSe:]FUNCtion:CUSTom:TCouple <*type*>[,<*range*>],(@<*ch_list*>).

The <*type*> parameter specifies the type of thermocouple wire so that the correct built-in table will be used for reference junction compensation. Reference junction compensation is based on the reference junction temperature at the time the custom channel is measured. For more information see "Thermocouple Reference Compensation" on page 142.

Custom Reference Temperature EU Conversions

The HP E1313/E1413 can measure reference junction temperatures using custom characterized RTDs and thermistors. The custom EU table generated for the individually characterized transducer is loaded to the appropriate channel(s) using the DIAGnostic:CUSTom:PIECewise command. Since the EU conversion from this custom EU table is to be considered the "reference junction temperature", the channel is linked to this EU table using the command [SENSe:]FUNCtion:CUSTom:REFerence [<*range*>,](@<*ch_list*>).

This command uses the custom EU conversion to generate the reference junction temperature as explained in the section "Thermocouple Reference Compensation" on page 142.

Creating Conversion Tables

To have conversion tables created for your application, contact your Hewlett-Packard System Engineer.

Summary

The following points describe the capabilities of custom EU conversion:

- A given channel only has a single active EU conversion table assigned to it. Changing tables requires loading it with a DIAGnostic:CUSTom:… command.
- The limit on the number of different custom EU tables that can be loaded in an HP E1313/E1413 is the same as the number of channels.
- Custom tables can provide the same level of accuracy as the built-in tables. Resistance measurements use a linear conversion table, and the temperature measurements use the piecewise conversion table.
- Consult with your HP System Engineer for custom EU tables for your application.

Compensating for System Offsets

System Wiring Offsets

The HP E1313/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 CALibration:TARE (@<*ch_list*>) command. The instrument will measure the voltage at each channel in <*ch_list*> and save those values in RAM as channel Tare constants.

Important Note for Thermocouples

- You must not use CALibration:TARE on field wiring that is made up of thermocouple wire. The voltage that a thermocouple wire pair generates cannot be removed by introducing a short anywhere between its junction and its connection to an isothermal panel (either the HP E1313/E1413's terminal module or a remote isothermal reference block). Thermal voltage is generated along the entire length of a thermocouple pair where there is any temperature gradient along that length. To CALibration:TARE thermocouple wire this way would introduce an unwanted offset in the voltage/temperature relationship for that thermocouple. If you inadvertently CALibration:TARE a thermocouple wire pair, see "Resetting CALibration:TARE" on page 139.
- You should use CALibration:TARE to compensate wiring offsets (copper wire, not thermocouple wire) between the HP E1313/E1413 and a remote thermocouple reference block. Disconnect the thermocouples and introduce copper shorting wires between each channel's HI and LO, then execute CALibration:TARE for these channels.

Residual Sensor Offsets

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

Operation

After CALibration: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 CALibration:TARE can remove is dependent on the A/D range. CALibration:TARE will choose the lowest range that can handle the existing offset voltage. The range that CALibration:TARE chooses will become the lowest usable range (range floor) for that channel. For any channel that has been "CALibration:TAREd" autorange will not go

below that range floor and selecting a manual range below the range floor will return an overload value (see table under FORMat[:DATA] in Chapter 5.

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 CALibration:TARE the module would return a reading of 0.1 Volt for channel 0. After CALibration: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. CALibration:TARE removes the *tare* portion leaving only the *net* signal value.

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

The tare calibration constants created during CALibration: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 CALibration:STORe TARE command.

Note

The HP E1313/E1413's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CALibration:STORe 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 CALibration:TARE

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

Special Considerations

Here are some things to keep in mind when using CALibration: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:

Table 4-6. Maximum CALibration: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 CALibration: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 CALibration:TARE. See the table under FORMat[:DATA] in Chapter 5 for more on values returned for overload readings.

Unexpected Channel Offsets or Overloads

This can occur when your HP E1313/E1413's Flash Memory contains CALibration:TARE offset constants that are no longer appropriate for its current application. Execute CALibration:TARE:RESet then *CAL? to reset the tare constants in RAM. Measure the affected channels again. If the problems go away, you can now reset the tare constants in Flash memory by executing CALibration:STORe TARE.

Detecting Open Transducers

Most of the HP E1313/E1413's SCPs provide a method to detect open transducers. When Open Transducer Detect (OTD) is enabled, the SCP injects a small current into the HIGH and LOW input of each channel. The polarity of the current pulls the HIGH inputs toward +17 volts and the LOW inputs towards -17 volts. If a transducer is open, measuring that channel will return an over-voltage reading. OTD is available on a per SCP basis. All eight channels of an SCP are enabled or disabled together. See Figure 4-9 for a simplified schematic diagram of the OTD circuit.

Notes

1. When OTD is enabled, the inputs have up to 0.2µA injected into them. If this current will adversely affect your measurement, but you still want to check for open transducers, you can enable OTD, make a single scan, check the CVT for bad measurements, then disable OTD and make your regular measurement scans. The specifications apply only when OTD is off.

2. When Filtering is enabled, allow 15 seconds for the filter capacitors to charge before checking for open transducers.

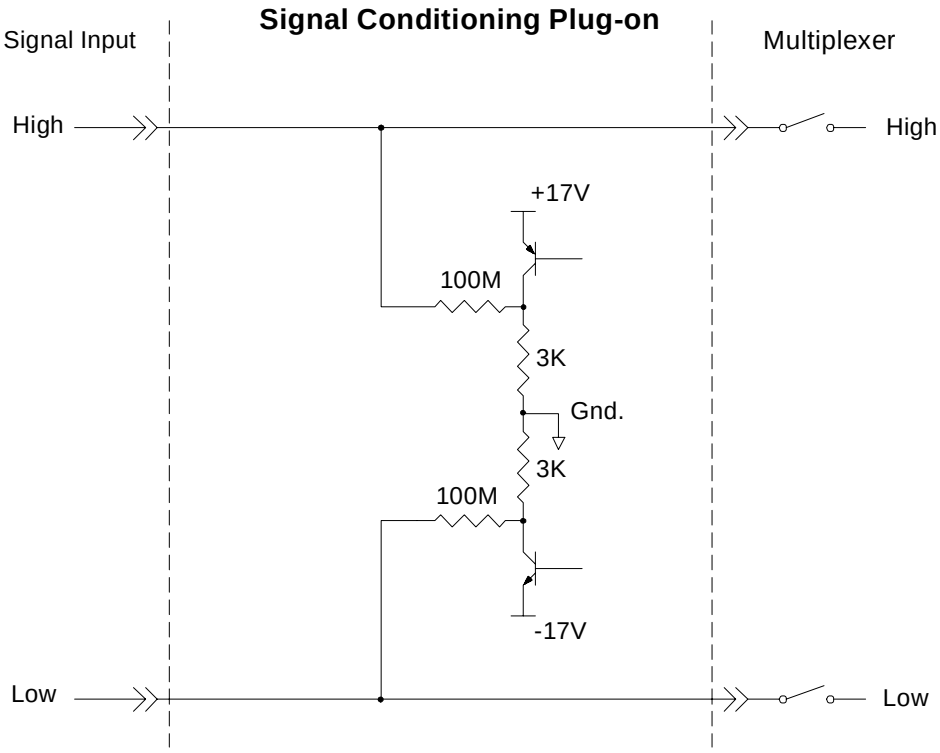


Figure 4-9. Simplified Open Transducer Detect Circuit

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

- The <*enable*> parameter can specify ON or OFF.
- An SCP is addressed when the <*ch_list*> parameter specifies a channel number contained on the SCP. The first channel on each SCP is 0, 8, 16, 24, 32, 40, 48, and 56.

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

```
DIAG:OTD  ON,(@100,116)          0 is on SCP 1 and 16 is on SCP 3.
```

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

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

Thermocouple Reference Compensation

The HP E1313/E1413 performs reference junction compensation automatically on all channels defined as thermocouple measurements by the [SENSe:]FUNCtion:TEMPerature TC command. There are two ways to do reference junction compensation in the HP E1313/E1413:

1. You can define the Reference Junction temperature to be a fixed value using the [SENSe:]REFerence:TEMPerature 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 [SENSe:]REFerence:TEMPerature 85.0 command. 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 predefined 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:]REFerence command, then including that channel in the scan list sequence before the thermocouple channels.

3. 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 2-11 on page 46 for the HP E1313 or Figure 2-12 on page 47 for the HP E1413.

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

   d. Put Channel 0 into the scan list before any thermocouple channels. Do this using the ROUTe:SEQuence:DEFine 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 108).

The SCPI Command sequence should be:

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

More On Autoranging

There are rare circumstances where your input signal can be difficult for the HP E1313/E1413 to autorange correctly. The module completes the range selection based on your input signal about 6 µsec before the actual measurement is made on that channel. If, during that period, your signal becomes greater than the selected range can handle, the module will return an overflow reading (±INFinity).

You can cure this problem by selecting a high enough manual range for that particular channel while leaving others in autorange (see the [SENSe:]FUNCtion:… commands in Chapter 5). You can also set the lowest allowable range that autorange can select for each channel by executing DIAGnostic:FLOor[:CONFigure] <*range*>,(@<*ch_list*>).

Reducing Settling Waits

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

Background

While the HP E1313/E1413 can autorange, measure, and convert a reading to engineering units as fast as once every 10 µs, measuring a high-level signal followed by a very low-level signal may require some extra settling time. As seen from the point of view of the HP E1313/E1413's Analog-to-Digital converter and its Range Amplifier, this situation is the most difficult to measure.

For example, lets look at two consecutive channels; the first measures a power supply at 15.5 volts, the next measures a thermocouple temperature. First the input to the Range Amplifier is at 15.5 volts (near its maximum) with any minute capacitances charged accordingly, then it immediately is switched to a thermocouple channel and down-ranged to its .0625 volt range. On this range, the resolution is now 1.91 µvolt per Least Significant Bit (LSB). Because of this sensitivity, the time to discharge these minute capacitances may have to be considered.

Checking for Problems

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

1. First run your system to make a record of its current measurement performance.
2. Then use the SAMPle:TIMer command to add a significant settling delay to every measurement in the scan list. Take care that the sample time multiplied by the number of channels in the scan list does not exceed the time between triggers.
3. Now run your system and look primarily for low-level channel measurements (like thermocouples) who's DC value changes somewhat. If you find channels that respond to this increase in sample period, you may also notice that these channels may return slightly quieter measurements as well. The extra sample period reduces or removes the affected channels coupling to the value of the channel measured just before it.
4. If you see some improvement, increase the sample period again and perform another test. When you increase the sample period and no improvement is seen, you have found the maximum settling delay that any single channel requires.
5. If the quality of the measurements does not respond to this increase in sample period, then your system was making good measurements without the added settling time.

Fixing the Problem

If your system scans fast enough with the increased sample period, your problem is solved. Your system is only running as fast as the slowest channel allows but if it is fast enough, that is acceptable. If, on the other hand, getting quality readings has slowed your scan rate too much, there are two other methods that will, either separately or in combination, have your system making good measurements as fast as possible.

Amplifier SCPs

Amplifier SCPs can remove the need to increase settling delays. How? Each gain factor of 4 provided by the SCP amplifier allows the Range Amplifier to be set one range higher and still provide the same measurement resolution. Amplifier SCPs for the HP E1413 are available with gains of 2, 8, 16, and 64. Lets return to our earlier example of a difficult measurement where one channel is measuring 15.5 volts on the 16 volt range, and the next a thermocouple on the .0625 range. If our thermocouple channel is amplified through an SCP with a gain of only 16, the Range Amplifier can be set to the 1 volt range. On this range the A/D resolution drops to around 31 μvolt per LSB so the stray capacitances discharging after the 15.5 volt measurement are now only one sixteenth as significant and thus reduce any required settling delay. Of course, for most thermocouple measurements,

we can use a gain of 64 and set the Range Amplifier to the 4 volt range. At this setting the A/D resolution for one LSB drops to about 122 µvolts and further reduces or removes any need for additional settling delay. This improvement is accomplished without any reduction of the overall measurement resolution.

Note

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

Adding Settling Delay for Specific Channels

This method adds settling time only to individual problem measurements rather then to all measurements as the SAMPle:TIMer command does.

With the HP E1313/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 (see the ROUTe:SEQuence:DEFine command in Chapter 5). 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 4-10 will illustrate this idea.

Of course you do not need to specify the optimum sequence for the scan list channels. You can simply place repeat channels in your existing scan list anywhere extra settling delay is needed.

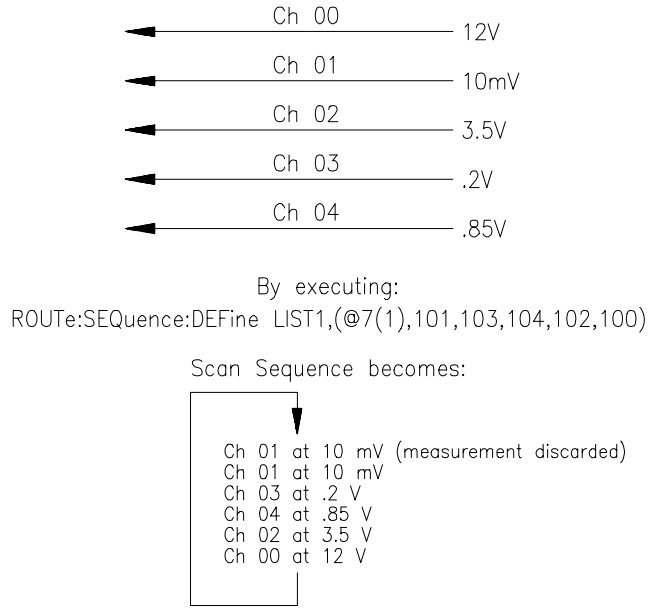


Figure 4-10. Recommended Scanning Sequence

Notes

Chapter 5
HP E1313/E1413 Command Reference

Using This Chapter

Except where noted, all references to the HP E1413 apply to the HP E1313. This chapter describes the **Standard Commands for Programmable Instruments** (SCPI) command set and the **IEEE-488.2 Common Commands** for the HP E1313/E1413.

Overall Command Index

SCPI Commands

Common Commands

Command Fundamentals

Commands are separated into two types: IEEE-488.2 Common Commands and SCPI Commands. The SCPI command set for the HP E1313/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

SCPI commands perform functions like starting a scan, 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:

```
ROUT:SEQ:POINT?  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 INITiate:IMMediate, 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

This section contains explanations and examples of parameter types you will see later in this chapter.

Parameter Types	Explanations and Examples
Numeric	Accepts all commonly used decimal representations of numbers including optional signs, decimal points, and scientific notation: 123, 123E2, -123, -1.23E2, .123, 1.23E-2, 1.23000E-01. Special cases include MIN, MAX, and INFinity. A parameter that represents units may also include a units suffix. These are: Volts; V, mv=10^{-3}, uv=10^{-6} 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 only one card, the card number (cc) will always be 1 and the channel number (nn) 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 222 for specific information.

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

(@<*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. 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;OUTPut:TTLTrg3 ON      or      TRIG:SOUR IMM;*TRG
```

Linking Multiple complete SCPI Commands. Use both a semicolon and a colon between the commands. For example:

```
OUTPut:TTLTrg2 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_type>,<sub_type>[,<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 Error -113, "Undefined header" 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.

Table 5-1. C-SCPI Data Types

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 E1313/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.

ABORt

The ABORt subsystem is a part of the HP E1313/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 wait-for-trigger state. If TRIGger:SOURce is IMMediate then the module resumes scanning. INITiate:CONTinuous 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 INITiate:CONTinuous OFF and check the Measuring bit (bit 4) with STATus:OPERation:CONDition?.
- **Related Commands:** INITiate:… commands, TRIGger:… commands
- ***RST Condition:** TRIGger:SOURce HOLD, INITiate:CONTinuous OFF

Usage

```
ABORt
```

If INITiated, goes to trigger idle state. If scanning, stops and goes to trigger idle state (will return to wait-for-trigger state if INITiate:CONTinuous is ON.

ARM

With the HP E1313/E1413, when the TRIGger:SOURce is set to TIMer, an ARM event must occur to start the timer. When INITiate:CONTinuous is set to ON and the TRIGger:SOURce is set to IMMediate (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 ARM:SOURce is related to TRIGger:SOURce and INITiate:CONTinuous and can, in certain settings, generate Error -221, "Settings conflict". See the note on page 266 for details.

The ARM command subsystem provides:

- An immediate software ARM (ARM[:IMMediate]).
- Selection of the ARM:SOURce <*source*> (<*source*> can be BUS, EXTernal, HOLD, IMMediate, SCP, or TTLTrg<*n*>) when TRIGger:SOURce is TIMer or the continuous mode is set (INITiate:CONTinuous ON and TRIGger:SOURce IMMediate).

Subsystem Syntax

```
ARM
    [:IMMediate]
    :SOURce <source>
    :SOURce?
```

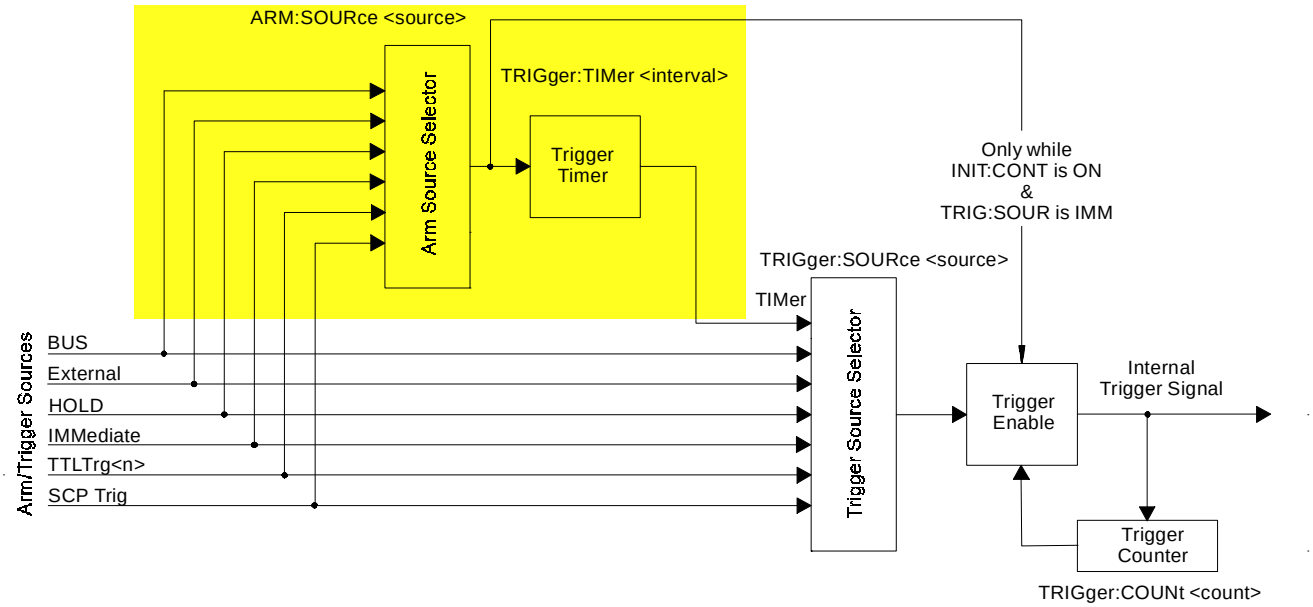


Figure 5-1. Logical Trigger Model

Figure 5-1 shows the overall logical model of the Trigger System.

ARM[:IMMediate]

ARM[:IMMediate] arms the trigger system when the module is set to the ARM:SOURce BUS or ARM:SOURce HOLD mode.

Comments

- ARM[:IMMediate] will cause Error -221, "Settings Conflict" unless ARM:SOURce is BUS or HOLD.
- **Related Commands:** ARM:SOURce
- ***RST Condition:** ARM:SOURce IMMediate

Usage

ARM:IMM	*After INIT, system is ready for trigger event.*
ARM	*Same as above (:IMM is optional).*

ARM:SOURce

ARM:SOURce ***<source>*** configures the ARM system to respond to the specified source.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*source*>	discrete (string)	BUS | EXT | HOLD | IMM | SCP | TTLTrg<*n*> (<*n*> = 1 through 7	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 the note about the ARM subsystem on page 160.

- The Arm and Trigger Sources can be changed while the module is scanning (but not in the continuous mode). This provides a way to stop triggering when you want to change the TRIGger:TIMer[:PERiod] <*trig_interval*>. To do this, execute:

```
ARM:SOURce IMMediate          Must be IMMediate when SOURce not TIMer.
TRIGger:SOURce BUS            To stop scanning.
        or
TRIGger:SOURce HOLD
TRIGger:TIMer <new_interval>  Change the timer interval.
TRIGger:SOURce TIMer          Return to timed scans.
```

- When TRIGger:SOURce is TIMer, TRIGger:TIMer:MODE is set to SYNC, and INITiate:CONTinuous is ON, an ARM event is required only to trigger the first scan. After that, the timer continues to run and the module goes to the wait-for-trigger state, ready for the next timer trigger. An INITiate:CONTinuous OFF command will return the module to the trigger idle state after the current scan is completed. See TRIGger:SOURce and TRIGger:TIMer:MODE for more details.

- While ARM:SOURce is IMMediate, you need only INITiate the trigger system to start a measurement scan.

- **Related Commands:** ARM[:IMMediate], ARM:SOURce?, INITiate[:IMMediate], INITiate:CONTinuous ON, TRIGger:SOURce

- ***RST Condition:** ARM:SOURce IMMediate

Usage

```
ARM:SOURce BUS      Arm with ARM command.
ARM:SOURce TTLTRG3  Arm with VXIbus TTLTRG3 line.
```

ARM:SOURce?

ARM:SOURce? returns the current arm source configuration. See the ARM:SOURce 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.
```

CALCulate

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
        :COUNt <n_readings>
        :COUNt?
        [:STATe] <enable>
        [:STATe]?
    :CLIMits
        :FAIL
            [:CUMulative]?
            :CURRent?
        :FLIMits
            [:CHANnels]
                [:CUMulative]?
                :CURRent?
            :POINts
                [CUMulative]?
                :CURRent?
    :LIMit
        :FAIL
            [:CUMulative]? (@<channel>)
            :CURRent? (@<channel>)
        :LOWer
            :DATA <lower_limit>,(@<ch_list>)
            :DATA? (@<channel>)
            [:STATe] <enable>,(@<ch_list>)
            [:STATe]? (@<channel>)
        [:STATe] <enable>,(@<ch_list>)
        [:STATe]? (@<channel>)
        :UPPer
            :DATA <upper_limit>,(@<ch_list>)
            :DATA? (@<channel>)
            [:STATe] <enable>,(@<ch_list>)
            [:STATe]? (@<channel>)
```

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 CALCulate:AVERage[: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 INITiate:CONTinuous is OFF,< *n_readings*> must be less than or equal to TRIGger:COUNt.
- **Related Commands:** CALCulate:AVERage:COUNt?, CALCulate:AVERage[:STATe]
- ***RST Condition:** CALCulate:AVERage:COUNt 2

Usage

```
CALC:AVER:COUN 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:** CALCulate:AVERage:COUNt, CALCulate:AVERage[:STATe]

Usage

```
CALC:AVER:COUN?
```

A subsequent enter statement will return the value currently set.

CALCulate:AVERage[:STATe]

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

Notes

1. When CALCulate:AVERage[:STATe] is ON, an individual channel number must appear in a scan list only once, and use of LISTL in not allowed.
2. When CALCulate:AVERage[:STATe] 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 CALCulate:AVERage:COUNt). The average of those measurements becomes the reading stored for that channel.
- **Related Commands:** CALCulate:AVERage:COUNt, CALCulate:AVERage[:STATe]?
- ***RST Condition:** CALCulate:AVERage:STATe 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 returned is **int16**.
- **Related Commands:** CALCulate:AVERage[:STATe], CALCulate:AVERage:COUNt

Usage

```
CALC:AVER?
```

A subsequent enter statement will return 0 or 1.

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 Operation Status Group and can generate a VXIbus interrupt.
- **Returned Value:** Numeric `0` or `1`. The C-SCPI type returned is **int16**.
- **Related Commands:** CALCulate:CLIMits:FLIMits[:CHANnels][:CUMulative]?, CALCulate:CLIMits:FLIMits:POINts:CURRent?, CALCulate:LIMit:… commands

Usage CALC:CLIM:FAIL:CUM?

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 Operation Status Group and can generate a VXIbus interrupt.
- **Returned Value:** Numeric `0` or `1`. The C-SCPI type returned is **int16**.
- **Related Commands:** CALCulate:CLIMits:FLIMits:…? commands, CALCulate:LIMit:… commands

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). This command 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.

Comments

- **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 returned is an **int16 array**).
- **Related Commands:** CALCulate:CLIMits:FAIL:…? commands, CALCulate:LIMit:… commands

Usage CALC:CLIM:FLIM:CHAN:CUM?

A subsequent 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). This command 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.

Comments

- **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 and the most significant channel bits last. The C-SCPI data type used to store the values is an array of 16-bit integer (**int16 array**).
- **Related Commands:** CALCulate:CLIMits:FAIL:…? commands, CALCulate:LIMit:… commands

Usage CALC:CLIM:FLIM:CHAN:CURR?

A subsequent 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 returned is **int16**.
- **Related Commands:** CALCulate:CLIMits:FLIMits:POINts:CURRent?

Usage

```
CALC:CLIM:FLIM:POIN?
```

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 returned is **int16**.
- **Related Commands:** CALCulate:CLIMits:FLIMits:POINts[:CUMulative]?

Usage

```
CALC:CLIM:FLIM:POIN:CURR?
```

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

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 returned is **int16**.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:LOWer:… commands, CALCulate:LIMit:UPPer:… commands

Usage

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

Return cumulative limit status for channel 2.

CALCulate:LIMit:FAIL:CURRent?

CALCulate:LIMit:FAIL:CURRent? (@<*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 returned is **int16**.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:LOWer:… commands, CALCulate:LIMit:UPPer:… commands

Usage

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

Return current limit status for channel 4.

CALCulate:LIMit:LOWer:DATA

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

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*lower_limit*>	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_limit*>.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:LOWer[:STATe], CALCulate:LIMit:UPPer:… commands
- ***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 returned is **float32**.
- **Related Commands:** CALCulate:LIMit:LOWer:DATA

Usage

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

Return the lower limit set for channel 6.

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
<*enable*>	discrete (string)	ON | OFF	none
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- If CALCulate:LIMit:LOWer[:STATe] is OFF, an overrange will not cause a limit exceeded status. If CALCulate:LIMit:LOWer[:STATe] is ON, an overrange will cause a limit exceeded status regardless of the setting of CALCulate:LIMit:LOWer:DATA.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:UPPer[:STATe]
- ***RST Condition:** CALCulate:LIMit:LOWer:STATe OFF

Usage

```
CALC:LIM:LOW:STAT 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`. The C-SCPI type returned is **int16**.
- **Related Commands:** CALCulate:LIMit:LOWer[:STATe]

Usage

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

Returns state of lower limit testing for channel 4.

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:** CALCulate:LIMit[:STATe]?
- ***RST Condition:** CALCulate:LIMit: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 returned is **int16**.
- **Related Commands:** CALCulate:LIMit[:STATe]

Usage

```
CALC:LIM:STAT? (@107)
```

Returns state of limit testing for channel 7.

CALCulate:LIMit:UPPer:DATA

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

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*upper_limit*>	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_limit*>.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:UPPer[:STATe], CALCulate:LIMit:LOWer:… commands
- ***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 returned is **float32**.
- **Related Commands:** CALCulate:LIMit:UPPer:DATA

Usage

```
CALC:LIM:UPP:DATA? (@107)
```
Returns the upper limit set for channel 7.

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
<*enable*>	discrete (string)	ON | OFF	none
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- If CALCulate:LIMit:UPPer[:STATe] is OFF, an overrange will not cause a limit exceeded status. If CALCulate:LIMit:UPPer[:STATe] is ON, an overrange will cause a limit exceeded status regardless of the setting of CALCulate:LIMit:UPPer:DATA.
- **Related Commands:** CALCulate:LIMit[:STATe], CALCulate:LIMit:LOWer[:STATe]
- ***RST Condition:** CALCulate:LIMit:UPPer:STATe OFF

Usage

```
CALC:LIM:UPP ON
```
Enables upper limit checking.

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`. The C-SCPI type returned is **int16**.
- **Related Commands:** CALCulate:LIMit:UPPer[:STATe]

Usage

```
CALC:LIM:UPP:STAT? (@104)
```

Returns the state of upper limit testing for channel 4.

CALibration

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 E1313/E1413's internal calibration sources. The known values are then sent to the HP E1313/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 (CALibration:CONFigure:…, CALibration:VALUE:…, and CALibration:STORe ADC). For an actual calibration example see the *HP E1313A/E1413C Service Manual*. Always execute *CAL? or a CALibration: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 autozero function in a conventional voltmeter. In the HP E1313/E1413, where channel scanning speed is of primary importance, this function is performed only when the CALibration: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 CALibration:SETup command.

   – **Channel Tare**
   This function (CALibration: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 (CALibration: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. CALibration:TARE automatically executes a *CAL?.

Subsystem Syntax

```
CALibration
    :CONFigure
        :RESistance
        :VOLTage <range>, ZERO | FSCale
    :SETup
    :SETup?
    :STORe <type>
    :TARE (@<ch_list>)
    :TARE?
        :RESet
    :VALue
        :RESistance <ref_ohms>
        :VOLTage <ref_volts>
    :ZERO?
```

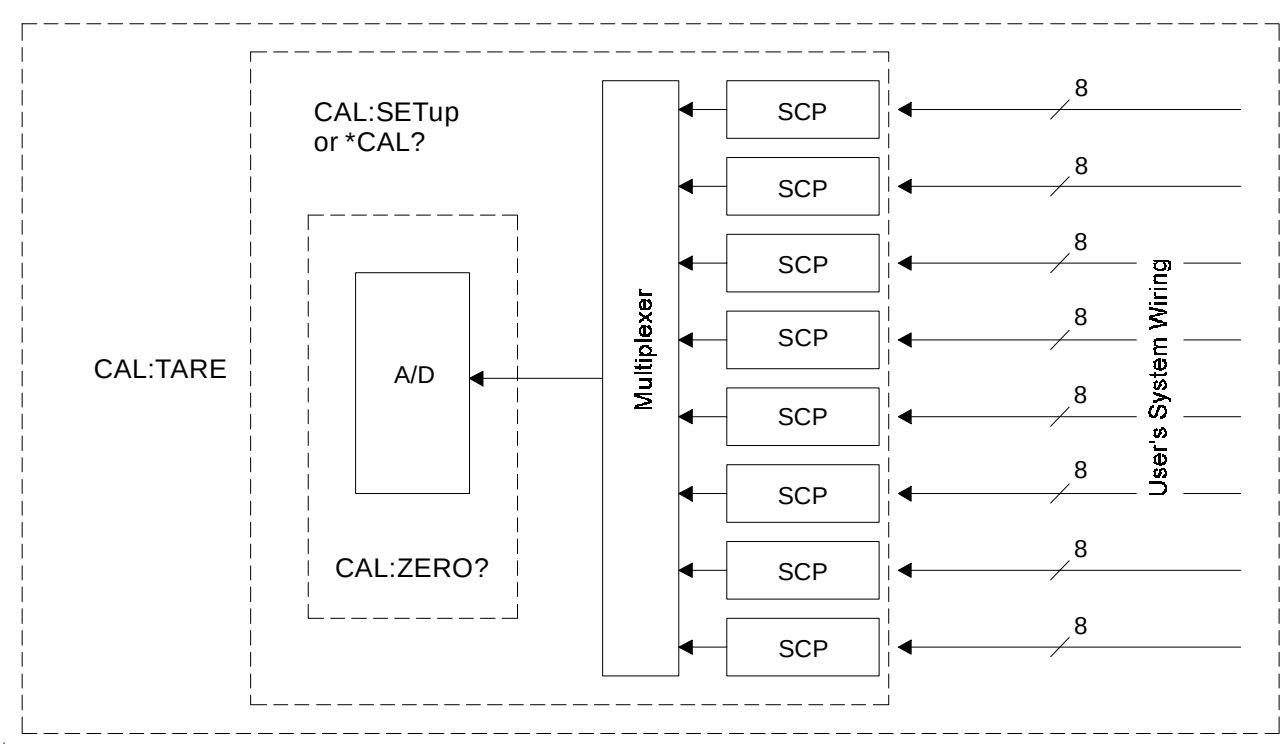


Figure 5-2. Levels of Working Calibration

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

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- CALibration:CONFigure:RESistance or CALibration:CONFigure:VOLTage commands *must be sent before* CALibration:VALue:RESistance or CALibration:VALue:VOLTage commands.
- The only CALibration command accepted after CALibration:CONFigure:RESistance is CALibration:VALue:RESistance.
- **Related Commands:** CALibration:VALue:RESistance, CALibration:STORe ADC

Command Sequence

CAL:CONF:RES	*Connects reference resistor to Calibration Bus.*
*OPC?	*Must wait for CAL:CONF:RES to complete.*
Now measure reference resistor with external DMM.	
CAL:VAL:RES *<measured value>*	*Send measured value to module.*
CAL:STOR 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)	.0625 | .25 | 1 | 4 | 16 (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: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, 16 VDC. <*range*> may be specified in millivolts (mv).
- CALibration:CONFigure:RESistance or CALibration:CONFigure:VOLTage commands *must be sent before* CALibration:VALue:RESistance or CALibration:VALue:VOLTage commands.
- The only CALibration command accepted after CALibration:CONFigure:VOLTage is CALibration:VALue:VOLTage.
- The CALibration:STORe command may be used to store the calibration constants in non-volatile memory.
- Perform ZERO before FSCale or Error +3050 will be generated.
- The FSCale output voltage of the calibration source will be greater than 90% of the nominal value for each range, except the 16 V range where the output is 10 V.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** CALibration:STORe ADC, CALibration:VALue:VOLTage

Command Sequence

CAL:CONF:VOLT .0625,ZERO	*Connect zero voltage reference to Calibration Bus.*
*OPC?	*Must wait for CAL:CONF:VOLT to complete.*
Now measure voltage with external DMM.	
CAL:VAL:VOLT <*measured value*>	*Inform HP E1413 of measured value.*
CAL:CONF:VOLT .0625,FSCale	*Connect full-scale voltage reference to Calibration Bus.*
*OPC?	*Must wait for CAL:CONF:VOLT to complete.*
Now measure voltage with external DMM.	
CAL:VAL:VOLT <*measured value*>	*Inform HP E1413 of measured value.*
Repeat these activities for each of 4 other ranges (.25, 1, 4, 16).	
.	
.	
.	
CAL:STOR ADC	*Store calibration constants in non-volatile memory (use 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. For more information see *CAL? on page 275.

Comments

- CALibration:SETup performs the same operation as the *CAL? command except that since it is not a query command it does not tie-up the C-SCPI driver waiting for response data from the instrument. If you have multiple HP E1313/E1413s in your system you can start a CALibration:SETup operation on each and then execute a CALibration:SETup? command to complete the operation on each instrument.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** CALibration:SETup?, *CAL?

Usage

```
CAL:SET                 Start SCP Calibration on first HP E1413.
:                       Start SCP Calibration on more HP E1413s.
CAL:SET                 Start SCP Calibration on last HP E1413.
CAL:SET?                Query for results from first 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. CALibration:SETup? returns the value only after the CALibration:SETup operation is complete.

Comments

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYSTem:ERRor?) See Error Messages in Appendix B. Also run *TST?.
-2	No results available	No *CAL? or CALibration:SETup done.

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

- **Related Commands:** CALibration:SETup, *CAL?

Usage See CALibration: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 CALibration:VALue for voltage and resistance, and stores these to Flash Memory. When <*type*> = TARE, the module stores the most recently measured CALibration:TARE channel offsets into Flash Memory.

Note The HP E1313/E1413's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CALibration:STORe 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.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*type*>	discrete (string)	ADC | TARE	none

Comments

- The Flash Memory Protect jumper (JM2201) must be set to the "PROG" position before executing this command (see Chapter 1).
- Channel offsets are compensated by the CALibration:TARE command even when not stored in the Flash Memory. There is no need to use the CALibration:STORe TARE command for channels which are recalibrated frequently.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** CALibration:VALue:RESistance, CALibration:VALue:VOLTage
- ***RST Condition:** Stored calibration constants are unchanged.

Usage

```
CAL:STOR  ADC          Store cal constants in non-volatile memory after A/D calibration.
CAL:STOR  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:STOR  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 CALibration: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 CALibration:STORe TARE command. For more information see "Compensating for System Offsets" on page 138.

Important Note for Thermocouples

- You must not use CALibration:TARE on field wiring that is made up of thermocouple wire. The voltage that a thermocouple wire pair generates cannot be removed by introducing a short anywhere between its junction and its connection to an isothermal panel (either the HP E1313/E1413's terminal module or a remote isothermal reference block). Thermal voltage is generated along the entire length of a thermocouple pair where there is any temperature gradient along that length. To CALibration:TARE thermocouple wire this way would introduce an unwanted offset in the voltage/temperature relationship for that thermocouple. If you inadvertently CALibration:TARE a thermocouple wire pair, see "Resetting CALibration:TARE" on page 139.

- You should use CALibration:TARE to compensate wiring offsets (copper wire, not thermocouple wire) between the HP E1313/E1413 and a remote thermocouple reference block. Disconnect the thermocouples and introduce copper shorting wires between each channel's HI and LO, then execute CALibration:TARE for these channels.

Parameters

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

Comments

- CALibration: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 +0.1 Volt offset with 0 Volts (a short) applied at the Unit Under Test (UUT). Before CALibration:TARE, the module would return a reading of 0.1 Volts for channel 0. After CALibration: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.

- Set Amplifier/Filter SCP gain before CALibration: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?.

- If Open Transducer Detect (OTD) is enabled when CALibration: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 CALibration:TARE, it should also wait 1 minute for settling.

- The maximum voltage that CALibration:TARE can compensate for is dependent on the range chosen and SCP gain setting. Table 5-2 lists these values.

Table 5-2. Maximum CALibration: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 CALibration:TARE command even when not stored in the Flash Memory. There is no need to use the CALibration:STORe TARE command for channels which are recalibrated frequently.

Note The HP E1313/E1413's Flash Memory has a finite lifetime of approximately ten thousand write cycles (unlimited read cycles). While executing CALibration:STORe 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.

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

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.

- **Related Commands:** CALibration:TARE?, CALibration:STORe TARE, DIAGnostic:CALibration:TARE[:OTDetect][:MODE]

- ***RST Condition:** Channel offsets are not affected by *RST.

Command Sequence

CAL:TARE @<*ch_list*>	*Correct channel offsets.*
CAL:TARE?	*Return the success flag from the CAL:TARE operation.*
CAL:STOR TARE	*Optional depending on necessity of long term storage.*

CALibration:TARE?

CALibration:TARE? returns a value to indicate the success of the last CALibration:TARE operation. CALibration:TARE? returns the value *only after* the CALibration:TARE operation is complete.

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYSTem:ERRor?) See Error Messages in Appendix B. Also run *TST?.
-2	No results available	Perform CALibration:TARE before CALibration:TARE?.

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

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

- **Related Commands:** CALibration:STORe TARE, DIAGnostic:CALibration:TARE[:OTDetect][:MODE]

Command Sequence

CAL:TARE @<*ch_list*>	*Correct channel offsets.*
CAL:TARE?	*Return the success flag from the CAL:TARE operation.*
CAL:STOR 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 CALibration:TARE:RESet affects the tare cal constants in RAM only. To reset the tare cal constants in Flash Memory, execute CALibration:TARE:RESet and then execute CALibration:STORe TARE.

Command Sequence

```
CAL:TARE:RES          Reset channel offsets.
CAL:STOR TARE         Optional if necessary to reset tare cal constants in Flash Memory.
```

CALibration:VALue:RESistance

CALibration:VALue:RESistance *<ref_ohms>* sends the just-measured value of the on-board reference resistor to the module for A/D calibration.

Parameters

Parameter Name	Parameter Type	Range of Value	Default Units
<*ref_ohms*>	numeric (float32)	7,500 ± 5%	ohms

Comments

- <*ref_ohms*> must be within 5% of the nominal reference resistor value (7,500 Ohms) 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 CALibration:CONFigure:RESistance command to configure the reference resistor for measurement at the Cal Bus connector.
- Use the CALibration:CONFigure:RESistance command prior to CALibration:VALue:RESistance or Error +3004, "illegal command" will occur.
- **Related Commands:** CALibration:CONFigure:RESistance, CALibration:STORe ADC

Command Sequence

```
CAL:CONF:RES
        Now measure voltage 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 just-measured 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

- The value sent must be for the currently configured range and output (ZERO or FSCale) as set by the previous CALibration:CONFigure:VOLTage <*range*>,<*zero_fs*> command. Full scale values must be within 10% of .0625, .25, 1, 4, or 10 (the voltage reference provides 10 VDC on the 16 V range).
- <*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 CALibration:CONFigure:VOLTage command prior to CALibration:VALue:VOLTage or Error +3004, "illegal command" will occur.
- Use the CALibration:CONFigure:VOLTage command to configure the on-board voltage source for measurement at the Cal Bus connector.
- **Related Commands:** CALibration:CONFigure:VOLTage, CALibration:STORe ADC

Command Sequence

```
CAL:CONF:VOLT  .0625,ZERO        Connect zero voltage reference to Calibration Bus.
*OPC?                            Must wait for CAL:CONF:VOLT to complete.
    Now measure voltage with external DMM.
CAL:VAL:VOLT  <measured value>   Send measured value to module.
    .
    .
    .
CAL:STOR  ADC                    Store calibration constants in non-volatile
                                 memory (use only at end of complete cal
                                 sequence).
```

CALibration:ZERO?

CALibration:ZERO? corrects Analog to Digital converter offset for any drift since the last *CAL? or CALibration:ZERO? command was executed.

Comments

- The CALibration: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 (SYSTem:ERRor?) 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?                    Returns 0 or -1.
```

DIAGnostic

The DIAGnostic subsystem allows you to perform special operations that are not standard in the SCPI language. This includes checking the current revision of the Control Processor's firmware, and that it has been properly loaded into Flash Memory.

Subsystem Syntax

```
DIAGnostic
    :CALibration
        :TARE
            [:OTDetect]
                [:MODE] <mode>
                [:MODE]?
    :CHECksum?
    :COMMand
        :SCPWRITE <reg_addr>,<reg_data>
    :CUSTom
        :LINear <table_range>,<table_block>,(@<ch_list>)
        :PIECewise <table_range>,<table_block>,(@<ch_list>)
        :REFerence
            :TEMPerature
    :FLOor <range>,(@<ch_list>)
        :DUMP
    :INTerrupt
        [:LINe] <intr_line>
        [:LINe]?
    :OTDetect
        [:STATe] <enable>,(@<ch_list>)
        [:STATe]? (@<channel>)
    :QUERy
        :SCPREAD? <reg_addr>
    :VERSion?
```

DIAGnostic:CALibration:TARE[:OTDetect][:MODE]

DIAGnostic:CALibration:TARE[:OTDetect][:MODE] *<mode>* sets whether Open Transducer Detect current will be turned off (the default mode) or left on during the CALibration:TARE operation.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*mode*>	boolean (uint 16)	0 | 1	volts

Comments

- When <*mode*> is set to 0 (the *RST Default), channels are tare calibrated with their OTD current off. When <*mode*> is 1, channels that have their OTD current on (DIAGnostic:OTDetect ON,(@<*ch_list*>)) are tare calibrated with their OTD current left on.

- By default (*RST) the CALibration:TARE? command will calibrate all channels with the OTD circuitry disabled. This is done for two reasons.
  1. Most users do not leave OTD enabled while taking readings.
  2. The CALibration:TARE operation takes much longer with OTD enabled.

  However, for users who intend to make readings with OTD enabled, setting DIAGnostic:CALibration:TARE:OTDetect:MODE 1 will force the CALibration:TARE command to perform the calibration with OTD enabled (on channels specified by the user with DIAGnostic:OTDetect:STATe).

- **Related Commands:** CALibration:TARE, CALibration:TARE?, DIAGnostic:CALibration:TARE[:OTDetect][:MODE]?, DIAGnostic:OTDetect[:STATe]

- ***RST Condition:** DIAGnostic:CALibration:TARE:OTDetect:MODE 0

Usage

Configure OTD on during CAL:TARE.

```
DIAG:CAL:TARE:MODE 1          Set mode for OTD to stay on.
CAL:TARE?                     Start channel tare cal.
```

DIAGnostic:CALibration:TARE[:OTDetect][:MODE]?

DIAGnostic:CALibration:TARE[:OTDetect][:MODE]? returns the currently set mode for controlling Open Transducer Detect current while performing CALibration:TARE operation.

Comments

- Returns a 0 when OTD current will be turned off during CALibration:TARE. Returns 1 when OTD current will be left on during CALibration:TARE operation. The C-SCPI type returned is **int16**.

- **Related Commands:** DIAGnostic:CALibration:TARE[:OTDetect][:MODE], DIAGnostic:OTDetect[:STATe], CALibration:TARE?

- ***RST Condition:** DIAGnostic:CALibration:TARE:OTDetect:MODE 0

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 returned 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 E1504 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 the Register Programming Section of your SCP Manual for parameter values.
- **Related Commands:** DIAGnostic:QUERy:SCPREAD?

Usage

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

For HP E1505 Current Source, set channel 7 on SCP 2 to 488 mA 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 E1313/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 below.	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 channels 16 - 23.
SENS:FUNC:CUST:LIN 1,1,(@116:123)      Link custom EU with channels 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 E1313/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 below.	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 channels 24 - 31 to
                                        HP E1413.
SENS:FUNC:CUST:PIEC  1,1,(@124:131)  Link custom EU with channels 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:PIEC <*table_range*>,<*table_block*>,(@<*ch_list*>)

Designate channel as reference.

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

Set up scan list sequence (channel 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:FLOor[:CONFigure]

DIAGnostic:FLOor[:CONFigure] <*range*>,(@<*ch_list*>) sets the lowest range that autorange can select for channels in <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*range*>	numeric (float32)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- There are rare circumstances when it is difficult for the HP E1313/E1413 to autorange correctly. The module selects the range depending on the input signal 6µS before the actual measurement is made on the selected channel. If, during that time period, the input signal becomes greater than the selected range can handle, the module returns an overflow reading (+INFinity). By locking-out that range (and lower range) for this channel, the module continues to autorange and avoid the overflow reading condition.

- The <*range*> parameter: The HP E1313/E1413 has the five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, specify the range value (for example, 4 selects the 4 VDC range). If you specify a value larger than one of the first four ranges, the HP E1313/E1413 selects the next higher range (for example, 4.1 selects the 16 VDC range). Specifying a value larger than 16 causes an error. Specifying 0 selects the lowest range (.0625 VDC).

- Once a channel's autorange floor is set by DIAGnostic:FLOor, it remains until reset by another DIAGnostic:FLOor command or by a *RST command.
- A channel with an autorange floor can be manually ranged below the floor ([SENSe:]FUNCtion:… commands). When the channel is returned to autorange, the autorange floor setting is still in effect.
- **Related Commands:** DIAGnostic:FLOor:DUMP, [SENSe:]FUNCtion:… commands.
- ***RST Condition:** DIAGnostic:FLOor 0,(@100:163)

Usage

```
DIAG:FLO 1,(@100,124)
```

Channel 0 and 24 can autorange no lower than 1 V range.

DIAGnostic:FLOor:DUMP

DIAGnostic:FLOor:DUMP sends the autorange floor settings for all 64 channels to the FIFO.

Comments

- The format of the values returned from the FIFO with a SENSe:DATA? command depends on the format chosen by the FORMat:DATA command.
- **Related Commands:** DIAGnostic:FLOor[:CONFigure], FORMat[:DATA], [SENSe:]FUNCtion:… commands
- ***RST Condition:** DIAGnostic:FLOor:DUMP places 0 values in the FIFO for all 64 channels.

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:** DIAGnostic:INTerrupt[:LINe]?
- **Power-on and *RST Condition:** DIAGnostic:INTerrupt:LINe 1

Usage

```
DIAG:INT:LIN 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 returned is **int16**.
- **Related Commands:** DIAGnostic:INTerrupt[:LINe]

Usage

```
DIAG:INT:LIN?                    Enter statement will return 0 through 7.
```

DIAGnostic:OTDetect[:STATe]

DIAGnostic:OTDetect[:STATe] <*enable*>,(@<*ch_list*>) enables and disables the HP E1313/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:** DIAGnostic:OTDetect[:STATe]?, DIAGnostic:CALibration:TARE[:OTDetect][:MODE]
- ***RST Condition:** DIAGnostic:OTDetect:STATe OFF

Note If OTD is enabled when *CAL?, or CALibration: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:STAT ON,(@100:107,115:123)   Select OTD for the first and third SCP
                                      (complete channel lists for readability only).
DIAG:OTD:STAT ON,(@100,115)           Same function as example above (only first
                                      channel of each SCP specified).
DIAG:OTD:STAT 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 returned is **int16**.
- **Related Commands:** DIAGnostic:OTDetect[:STATe]

Usage

```
DIAG:OTD:STAT? (@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 E1504 Breadboard SCP.

Parameters

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

Comments

- See the Register Programming Section of your SCP Manual for parameter values.
- **Returned Value:** Returns numeric register value. The C-SCPI type returned is **int32**.

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:** Examples of the response string format:
  `HEWLETT-PACKARD,E1413,US34000478,A.04.00,Thu Aug 5 9:38:07 MDT 1994`
  `HEWLETT-PACKARD,E1313_32CH,US35000134,A.04.00, Thu Aug 25 15:11:20 MDT 1994`
- The C-SCPI type returned is **string**.
- **Related Commands:** *IDN?

Usage

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

FETCh?

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 INITiate:… 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 and Response Data" on page 156 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[:DATA] 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:** FORMat[:DATA] and MEMory:… commands.
- ***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 HP E1413 for scanning)

TRIG:SOUR  IMM                Let unit trigger on INIT.
INIT:IMM                      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.
FETC?
```

Note When using the MEMory subsystem, the module must be triggered before executing the INITiate:IMMediate command (as shown above) unless you are using an external trigger (EXTernal trigger). When using EXTernal trigger, the trigger can occur at any time.

FORMat

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 and Response Data" on page 156 of this chapter. For REAL,32, readings are 4 bytes in length (C-SCPI type returned is **float32 array**). For REAL,64 and PACK,64, readings are 8 bytes in length (C-SCPI type returned 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 returned is **string array**).

Note *TST? leaves the instrument in its power-on reset state. This means that the ASC,7 data format is set even if you had it set to something else before executing *TST?. If you need to read the FIFO for test information, set the format after *TST? and before reading the FIFO.

- **Related Commands:** [SENSe:]DATA:FIFO:… ? commands, [SENSe:]DATA:CVTable?, MEMory:… commands, 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

```
FORM:DATA  REAL          Set format to IEEE 32-bit Floating Point.
FORM:DATA  REAL,64       Set format to IEEE 64-bit Floating Point.
FORM:DATA  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 returned is **string, int16**.
- **Related Commands:** FORMat[:DATA]
- ***RST Condition:** ASCII, 7

Usage

```
FORM:DATA?                    Returns REAL,+32 | REAL,+64 |
                              PACK,+64 | ASC,+7
```

INITiate

The INITiate command subsystem moves the HP E1313/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, or the List-of-Lists (specified by ROUTe:SCAN LIST1 - LIST4 or LISTL) 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 <enable>
    [:IMMediate]
```

INITiate:CONTinuous

INITiate:CONTinuous *<enable>*. ON changes the trigger system from the Trigger Idle State to the Initiated State. An arm/trigger event executes one or more scans, then the module returns to the Initiated State ready for the next trigger. INITiate:CONTinuous OFF cancels the CONTinuous mode.

Note INITiate:CONTinuous is related to TRIGger:SOURce and ARM:SOURce and can, in certain settings, generate Error -221,"Settings conflict". See the note on page 266 for details.

Parameters

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

Comments

- The term "Continuous Mode" means that the module is scanning with TRIGger:SOURce IMMediate *and* INITiate:CONTinuous ON. After an ARM event, the module executes a single Scan List continuously until an INITiate:CONTinuous OFF is received. Continuous Mode provides the fastest scan rate. See Figure 5-6 on page 267.
- Executing INITiate:CONTinuous ON clears the FIFO and Current Value Table.
- If a trigger event is received before the instrument is initiated, Error -211, "Trigger ignored" 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 Error +3012, "Trigger too fast" is generated.

- Sending the ABORt command will immediately send the trigger system back to its Trigger Idle state and terminate any scan in progress. If INITiate:CONTinuous is ON, however, it will immediately return to the Wait-for-Trigger state. If TRIGger:SOURce is IMMediate also, then the module will resume scanning. Use INITiate:CONTinuous OFF to stop after the current scan is complete. You can check the "Measuring" bit (bit 4) with STATus:OPERation:CONDition? to determine when the scan has completed.
- Sending INITiate:IMMediate while the system is still in the Wait-for-Trigger state (already INITiated) will cause Error -213, "Init ignored".
- **Related Commands:** ABORt, TRIGger[:IMMediate], TRIGger:SOURce
- ***RST Condition:** Trigger system is in the Trigger Idle state.

Usage

```
INIT:CONT  ON        Return to Initiated State after each trigger.
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

- INITiate:IMMediate clears the FIFO and Current Value Table.
- If a trigger event is received before the instrument is initiated, Error -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 Error +3012, "Trigger too fast" error is generated.
- Sending INITiate:IMMediate while the system is still in the Wait-for-Trigger state (already INITiated) will cause Error -213,"Init ignored".
- Sending the ABORt command will immediately send the trigger system to the Trigger Idle state and terminate any scan in progress.
- **Related Commands:** ABORt, TRIGger
- ***RST Condition:** Trigger system is in the Idle state.

Usage

```
INIT:IMM             Both versions same function.
INITiate:IMMediate
```

INPut

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] <enable,(@<channel>)
            [:STATe]? (@<channel>)
    :GAIN <chan_gain>,(@<ch_list>)
    :GAIN? (@<channel>)
    :LOW <wvolt_type>,(@<ch_list>)
    :LOW? (@<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 Comments below. MIN | MAX	Hz
<ch_list>	channel list (string)	100 - 163	none

Comments

- *<cutoff_freq>* may be specified in kiloHertz (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 INPut:FILTer:LPASs:STATe OFF command.
- Sending a value greater than the SCP's highest cutoff frequency or less than the SCP's lowest cutoff frequency generates Error -222, "Data out of range" error.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** INPut:FILTer[:LPASs]:FREQuency?, INPut:FILTer[:LPASs][:STATe]
- ***RST Condition:** Set to MIN.

Usage

```
INP:FILT:FREQ  100,(@100:119)      Set cutoff frequency of 100 Hz for first 20 channels.
INP:FILT:FREQ  2,(@155)            Set cutoff frequency of 2 Hz for channel 55.
```

INPut:FILTer[:LPASs]:FREQuency?

INPut:FILTer[:LPASs]:FREQuency? (@*<channel>*) returns the cutoff frequency currently set for *channel*. Non-programmable SCP channels may be queried to determine their fixed cutoff frequency.

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 INPut:FILTer:LPASs:FREQuency command. The C-SCPI type returned is **float32**.
- **Related Commands:** INPut:FILTer[:LPASs]:FREQuency, INPut:FILTer[:LPASs][:STATe]
- ***RST Condition:** MIN

Usage

```
INP:FILT:LPAS:FREQ?  (@155)        Check cutoff frequency on channel 55.
INP:FILT:LPAS:FREQ?  (@100)        Check cutoff frequency 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.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- ***RST Condition:** ON

Usage

```
INP:FILT:LPAS:STAT  ON,(@115,117)
```
Channels 115 and 117 return to previously set (or default) cutoff frequency.

```
INP:FILT:LPAS:STAT  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

- <*channel*> must specify a single channel only.
- **Returned Value:** Numeric value either 0 (off or "pass-through") or 1 (on). The C-SCPI type returned is **int16**.

Usage

```
INP:FILT:LPAS:STAT? (@115)
```
Enter statement returns either 0 or 1.

```
INP:FILT? (@115)
```
Same as above.

INPut:GAIN

INPut:GAIN <*chan_gain*>,(@<*ch_list*>) sets the channel gain on programmable amplifier Signal Conditioning Plug-ons.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*chan_gain*>	numeric (float32) discrete (string)	See Comments below | 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 Error -222, "Data out of range".
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** INPut:GAIN?
- ***RST Condition:** Gain set to MIN.

Usage

```
INP:GAIN  8,(@100:119)      Set gain of 8 for first 20 channels.
INP: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, INPut:GAIN? will return the nominal as-designed gain for that channel.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Returned Value:** Numeric value as set by the INPut:GAIN command. The C-SCPI type returned is **float32**.

- **Related Commands:** INPut:GAIN
- ***RST Condition:** Gain set to 1.

Usage

```
INP:GAIN? (@105)        Check gain on channel 5.
INP:GAIN? (@100)        Check gain on channel 0.
```

INPut:LOW

INPut:LOW <*wvolt_type*>,(@<*ch_list*>) controls the connection of input LO at a Strain Bridge SCP channel specified by <*ch_list*>. LO can be connected to the Wagner Voltage ground or left floating.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*wvolt_type*>	discrete (string)	FLOat | WVOLtage	none
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- **Related Commands:** INPut:LOW?
- ***RST Condition:** INPut:LOW FLOat (all HP E1511 Transient Strain SCP channels)

Usage

```
INPut:LOW WVOL,(@100:103,116:119)    Connect LO of channels 0 through 3 and 16
                                     through 19 to Wagner Ground.
```

INPut:LOW?

INPut:LOW? (@<*channel*>) returns the LO input configuration for the channel specified by <*channel*>.

Parameters

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

Comments

- <*channel*> must specify a single channel only.
- **Returned Value:** Returns FLO or WV. The C-SCPI type returned is **string**.
- **Related Commands:** INPut:LOW

Usage

```
INP:LOW? (@103)        Enter statement will return either FLO or
                       WV for channel 3.
```

MEMory

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 <enable>
        :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 HP E1413 for scanning)

TRIG:SOUR IMM                  Let unit trigger on INIT.
INIT:IMM
*OPC?                          Program execution remains here until VME
                               memory is full or the HP E1413 has stopped
                               taking readings.
FORMat:DATA REAL,64            Affects only the return of data.
FETC?                          Return data from VME memory.
```

Note When using the MEMory subsystem, the module must be triggered before executing the INITiate:IMMediate command (as shown above) unless you are using an external trigger (EXTernal trigger). When using an EXTernal 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 MEMory:VME:ADDRess not executed) is 240000_{16}.
- <*A24_address*> may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- **Related Commands:** MEMory… commands, FORMat:… commands, and FETCh?
- ***RST Condition:** VME memory address starts at 200000_{16}. When using an HP E1405B or HP E1406A 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… commands, FORMat:… commands, and FETCh?

Usage

```
MEM:VME:ADDR?
```

Returns the address of the VME memory card.

MEMory:VME:SIZE

MEMory:VME:SIZE *<mem_size>* specifies the number of bytes of VME memory to allocate for additional reading storage.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*mem_size*>	numeric	To limit available VME memory	none

Comments

- This command is only available in systems using an HP E1405B or HP E1406A Command Module.
- <*mem_size*> may be specified in decimal, hex (#H), octal (#Q), or binary (#B).
- <*mem_size*> should be a multiple of four (4) to accommodate 32 bit readings.
- **Related Commands:** MEMory… commands, FORMat:… commands, and FETCh?
- ***RST Condition:** MEMory: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 value.
- **Related Commands:** MEMory… commands 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 INITiate:IMMediate command does not terminate until data acquisition stops or VME memory is full.
- **Related Commands:** MEMory… commands and FETCh?
- ***RST Condition:** MEMory:VME:STATe OFF

Usage

```
MEM:VME:STAT 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 value either `1` or `0`.
- **Related Commands:** MEMory… commands and FETCh?

Usage

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

OUTPut

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] <enable>,(@<ch_list>)
        [:STATe]? (@<channel>)
    :SHUNt <enable>,(@<ch_list>)
    :SHUNt? (@<channel>)
    :TTLTrg
        :SOURce <trig_source>
        :SOURce?
    :TTLTrg<n>
        [:STATe] <ttltrg_cntrl>
        [:STATe]?
    :VOLTage
        :AMPLitude <amplitude>,(@<ch_list>)
        :AMPLitude? (@<channel>)
```

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.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** *CAL?, OUTPut:CURRent:AMPLitude?
- ***RST Condition:** MIN

Usage

```
OUTPut:CURRent:AMPLitude  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, Error +3007, "Invalid signal conditioning module" is generated.
- **Returned Value:** Numeric value of amplitude set. The C-SCPI type returned is **float32**.
- **Related Commands:** OUTPut:CURRent:AMPLitude

Usage

```
OUTP:CURR:AMPL?  (@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

- OUTPut:CURRent:STATe 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 OUTPut:CURRent:AMPLitude command.
- OUTPut:CURRent:STATe is most commonly used to turn off excitation current to four-wire resistance (and resistance temperature device) circuits during execution of CALibration:TARE for those channels.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** OUTPut:CURRent:AMPLitude, CALibration:TARE
- ***RST Condition:** OUTPut:CURRent:STATe OFF (all channels)

Usage

```
OUTP:CURR:STAT 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. The C-SCPI type returned is **uint16**.
- **Related Commands:** OUTPut:CURRent[:STATe], OUTPut:CURRent:AMPLitude

Usage

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

OUTPut:SHUNt[:STATe]

OUTPut:SHUNt[:STATe] <*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)	1 | 0 | ON | OFF	none
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- If <*ch_list*> specifies a non-strain SCP, Error +3007, "Invalid signal conditioning module" is generated.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** [SENSe:]FUNCtion:STRain:… commands and [SENSe:]STRain:… commands.
- ***RST Condition:** OUTPut:SHUNt:STATe 0 on all Strain SCP channels.

Usage

```
OUTP:SHUN:STAT 1,(@116:119)
```

Add shunt resistance at channels 16 through 19.

OUTPut:SHUNt[:STATe]?

OUTPut:SHUNt[:STATe]? (@<*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, Error +3007, "Invalid signal conditioning module" is generated.
- **Returned Value:** Returns 1 or 0. The CSCPI type returned is **uint16**.
- **Related Commands:** OUTPut:SHUNt[:STATe]

Usage

```
OUTP:SHUN:STAT? (@116)
```

Check status of shunt resistance on channel 16.

OUTPut:TTLTrg:SOURce

OUTPut:TTLTrg:SOURce ***<trig_source>*** selects the internal source of the trigger event that will operate the VXIbus TTLTRG lines.

HP E1313 Note Most B-Size mainframes do not support 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 TRIGger: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 TRIGger:SOURce <*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 TRIGger: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 OUTPut:TTLTrg:SOURce FTRigger and OUTPut:TTLTrg<*n*>:STATe ON) a second HP E1413 (set to TRIGger:SOURce TTLTrg<*n*>). The second module will be triggered in the ratio

$$\frac{1}{TRIGger{:}COUNt_{module\#1}}$$

  For an example of this arrangement, see "Timer Based Scans at Different Rates" on page 118.

- **Related Commands:** OUTPut:TTLTrg<*n*>[:STATe], OUTPut:TTLTrg:SOURce?, TRIGger:SOURce, TRIGger:COUNt

- ***RST Condition:** OUTPut:TTLTrg:SOURce TRIGger

Usage

```
OUTP:TTLT:SOUR LIM          Toggle TTLTRGn line when limit test exceeded.
```

OUTPut:TTLTrg:SOURce?

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

Comments

- **Returned Value:** Discrete, one of; `TRIGger`, `FTRigger`, `SCPlugon`, or `LIMit`. The C-SCPI type returned is **string**.
- **Related Commands:** OUTPut:TTLTrg:SOURce

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, OUTPut:TTLTRG4 or OUTPut:TTLT4 for VXIbus TTLTRG line 4.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*n*>	numeric	1 through 7	none
<*ttltrg_cntrl*>	discrete (string)	ON | OFF	none

Comments

- Only one VXIbus TTLTRG line can be enabled simultaneously.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** ABORt, INITiate:… commands, TRIGger:… commands
- ***RST Condition:** OUTPut:TTLTrg<0 through 7> OFF

Usage

```
OUTPut:TTLTrg2:STATe ON
```
Enable TTLTRG2 line to source a trigger.

```
OUTP:TTLT7:STAT ON
```
Enable TTLTRG7 line to source a trigger.

OUTPut:TTLTrg<*n*>[:STATe]?

OUTPut:TTLTrg<*n*>[:STATe]? returns the current state for TTLTRG line <*n*>.

Comments

- **Returned Value:** Returns `1` or `0`. The C-SCPI type returned is **int16**.
- **Related Commands:** OUTPut:TTLTrg<*n*>[:STATe]

Usage

```
OUTP:TTLT2?
```
See if TTLTRG2 line is enabled (returns 1 or 0).

```
OUTP:TTLT7:STAT?
```
See if TTLTRG7 line is enabled.

OUTPut:VOLTage:AMPLitude

OUTPut:VOLTage:AMPLitude <*amplitude*>,(@<*ch_list*>) sets the excitation voltage on programmable Strain Bridge Completion SCPs pointed to by <*ch_list*> (the HP E1511 Transient Strain SCP, for example).

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*amplitude*>	numeric (float32)	MIN | 0 | 1 | 2 | 5 | 10 | MAX	none
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- To turn off excitation voltage (when using external voltage source), program <*amplitude*> to 0.
- **Related Commands:** OUTPut:VOLTage:AMPLitude?
- ***RST Condition:** MIN (0)

Usage

```
OUTP:VOLT:AMPL 5,(@116:119)
```

Set excitation voltage for channels 16 through 19.

OUTPut:VOLTage:AMPLitude?

OUTPut:VOLTage:AMPLitude? (@<*channel*>) returns the current setting of excitation voltage for the channel specified by <*channel*>.

Comments

- <*channel*> must specify a single channel only.
- **Returned Value:** Numeric, one of `0`, `1`, `2`, `5`, or `10`. The C-SCPI type returned is **float32**.
- **Related Commands:** OUTPut:VOLTage:AMPLitude

Usage

```
OUTP:VOLT:AMPL? (@103)
```

Returns current setting of excitation voltage for channel 3.

ROUTe

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 <scan_list>
    :SEQuence:
        :DEFine <scan_list>,(@<ch_list>)
        :DEFine? <scan_list>[,<mode>]
        :POINts? <scan_list>
```

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 | LIST2 | LIST3 | LIST4 | LISTL	none

Comments

- When ROUTe:SCAN is LISTL, the module will automatically sequence through the scan lists specified by ROUTe:SEQuence:DEFine LISTL,(@<*ch_list*>).

LISTL Notes

1. When ROUTe:SCAN selects LISTL (List-of-Lists), each scan list it specifies must contain at least 6 channels.
2. LISTL can only be selected while module is in Trigger Idle State (not INITiated).
3. When CALCulate:AVERage:STATe 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 Wait-for-Trigger State. If ROUTe:SCAN LIST1-LIST4 is executed after this point, it will become effective for the next scan. If INITiate:CONTinuous is ON and TRIGger:SOURce is IMMediate, ROUTe:SCAN will generate an error. Refer to Figure 5-3 for scanning event sequence.
- **Related Commands:** ABORt, INITiate:… commands, ROUTe:SEQuence:DEFine
- ***RST Condition:** ROUTe:SCAN LIST1

Usage

```
ROUT:SCAN LIST3
```

The next scan list executed will be LIST3.

NOTES:
1. ROUTe:SCAN LISTL can only be executed while in Trigger Idle State.

2. ROUTe:SCAN LIST1-LIST4 can be executed any time.

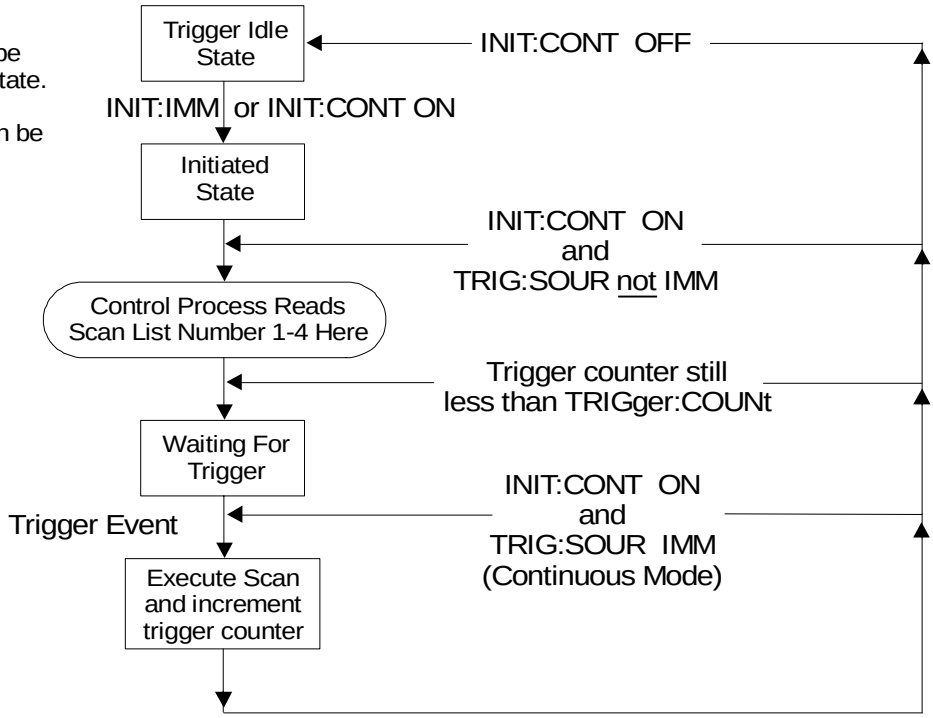


Figure 5-3. Event Sequence for ROUTe:SCAN

ROUTe:SEQuence:DEFine

ROUTe:SEQuence:DEFine <*scan_list*>,(@<*ch_list*>) defines channel content and sequence for Scan List LIST<*n*>, or defines 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 | LIST2 | LIST3 | LIST4 | LISTL | ALL	none
<*ch_list*>	channel list (string)	for LIST1 | 2 | 3 | 4 or ALL, use 100 - 163 | <*cc*> (@00:63) for LISTL, use LIST1 | 2 | 3 | 4	none

Comments

- Channels can be specified in any order.
- When CALCulate:AVERage:STATe is OFF, channels can be specified multiple times in an individual Scan List. When CALCulate:AVERage:STATe is ON, each channel must appear only once in an individual Scan List.
- A channel list may contain as many as 1,024 entries.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- When ROUTe:SCAN selects LIST1 through LIST4, the specified scan list can contain as few as two channels (three channels when an HP E1510/E1511 is installed). When ROUTe:SCAN selects LISTL, each Scan List specified in the List-of-Lists must contain at least 6 entries or else Error +3001, "Illegal while continuous" will occur.
- 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 ROUTe:SEQuence:DEFine 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 do not store result in either FIFO or CVT. Use as dummy channel set.

* Limit Checking (CALCulate:LIMit:…) cannot 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:SEQuence:DEFine 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 ROUTe:SEQuence:DEFine, use [SENSe:]FUNCtion:… commands to set each channel to a specific function and range.

- **Related Commands:** ROUTe: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 | LIST2 | LIST3 | LIST4 | LISTL	none
<*mode*>	discrete (string)	CHANnel | MODifier	none

Comments

- The default for <*mode*> is CHAN.
- When <*scan_list*> is LIST1 - LIST4 and <*mode*> is CHAN, ROUTe:SEQuence:DEFine? returns the sequence of channels assigned to <*scan_list*>.
- When <*scan_list*> is LIST1 - LIST4 and <*mode*> is MOD, ROUTe:SEQuence:DEFine? 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 do not store result in either FIFO or CVT. Use as dummy channel set.

* Limit Checking (CALCulate:LIMit:...) cannot be performed on channels that are not converted to Engineering Units.

- **Returned Value:** Returned values are as a comma separated list.
- **Related Commands:** ROUTe:SEQuence:DEFine, ROUTe: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, this command 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 | LIST2 | LIST3 | 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 ROUTe:SEQuence:DEFine command) of 3, 4 or 7 can be used which does not place the readings in FIFO.
- **Returned Value:** Numeric. The C-SCPI type returned is **int16**.
- **Related Commands:** ROUTe:SEQuence:DEFine
- ***RST Condition:** Scan Lists contain zero channels.

Usage

```
ROUT:SEQ:POIN?  LIST3
```

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

SAMPle

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  <scan_list>,<interval>
    :TIMer?  <scan_list>
```

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 SAMPle:TIMer 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 | LIST2 | LIST3 | LIST4 | LISTL | ALL	none
<*interval*>	numeric (float 32) (string)	1.0E-5-5 to 32.768E-3 | MIN | MAX	seconds

Comments

- The minimum <*interval*> is 10 µ seconds. The resolution for <*interval*> is .5 µs. <*interval*> may be specified in milliseconds (ms), or microseconds (us).

Note If the A/D filter is on (SENSe:FILTer:LPASs:STATe ON) then the minimum *interval* is 145 µsec.

- 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 Error +3012, "Trigger too fast" will be generated.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** SAMPle:TIMer?
- ***RST Condition:** Sample Timer for all Channel Lists set to 1.0E-5 seconds.

Usage

```
SAMP:TIM  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 | LIST2 | LIST3 | LIST4 | LISTL |	none

Comments

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

Usage

```
SAMP:TIM? LIST4
```

Check the interval between channel measurements for scan list 4.

[SENSe:]

Subsystem Syntax

```
[SENSe:]
    DATA
        :CVTable? (@<ch_list>)
            :RESet
        :FIFO
            [:ALL]?
            :COUNt?
                :HALF?
            :HALF?
            :MODE <mode>
            :MODE?
            :PART? <n_readings>
            :RESet
    FILTer
        [:LPASs]
            [:STATe] <enable>
            [: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>,][<range>,](@<ch_list>)
        :TEMPerature <degrees_celsius>
    STRain
        :EXCitation <excite_v>,(@<ch_list>)
        :EXCitation? (@<channel>)
        :GFACtor <gage_factor>,(@<ch_list>)
        :GFACtor? (@<channel>)
        :POISson <poisson_ratio>,(@<ch_list>)
        :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 (string)	100 - 163	none

Comments

- This command 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 INITiate:IMMediate 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 returned 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 and Response Data" section on page 156 of this chapter. For REAL,32, each reading is 4 bytes in length (the C-SCPI data type returned is a **float32 array**). For REAL,64 and PACK,64, each reading is 8 bytes in length (the C-SCPI data type returned 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 SENSe: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 200 for actual values for each data format).

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

Usage

```
SENS:DATA:CVT? (@100:163)      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).
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** [SENSe:]DATA:CVTable?
- ***RST Condition:** SENSe:DATA:CVTable:RESet

Usage

```
SENS:DATA:CVT:RES                    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

- SENSe:DATA:FIFO:ALL? 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 SENSe: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 returned 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 and Response Data" section on page 156 of this chapter. For REAL,32, each reading is 4 bytes in length (the C-SCPI data type returned is a **float32 array**). For REAL,64 and PACK,64, each reading is 8 bytes in length (the C-SCPI data type returned is a **float64 array**).

Note Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see the table on page 200 for actual values for each data format).

- **Related Commands:** [SENSe:]DATA:FIFO:HALF?

- ***RST Condition:** FIFO is empty.

Usage

```
SENSe: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

- SENSe:DATA:FIFO:COUNt? is used to determine the number of readings to acquire with the SENSe:DATA:FIFO:PART? command.
- **Returned Value:** Numeric `0` through `65,024`. The C-SCPI type returned is **int32**.
- **Related Commands:** [SENSe:]DATA:FIFO:PART?
- ***RST Condition:** FIFO empty.

Usage

```
SENSe: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

- SENSe: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 returned is **int16**.
- **Related Commands:** [SENSe:]DATA:FIFO:HALF?
- ***RST Condition:** FIFO empty.

Command Sequence

```
SENSe:DATA:FIFO:COUNt:HALF?
SENSe:DATA:FIFO:HALF?
```
Poll FIFO for half-full status.
Returns 32768 readings.

[SENSe:]DATA:FIFO:HALF?

[SENSe:]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 SENSe:DATA:FIFO:HALF? command and a read statement often enough to keep up with the scan reading rate.
- Use the SENSe: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 returned 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 the "Arbitrary Block Program and Response Data" section on page 156 of this chapter. For REAL,32, each reading is 4 bytes in length (the C-SCPI data type returned is a **float32 array**). For REAL,64 and PACK,64, each reading is 8 bytes in length (the C-SCPI data type returned is a **float64 array**).

Note Channel readings which are a positive overvoltage return IEEE +INF and a negative overvoltage return IEEE -INF (see the table on page 200 for actual values for each data format).

- **Related Commands:** [SENSe:]DATA:FIFO:COUNt:HALF?
- ***RST Condition:** FIFO buffer is empty.

Command Sequence

```
SENSe:DATA:FIFO:COUNt:HALF?     Poll FIFO for half-full status.
SENSe: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.

Parameters

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

Comments

- In BLOCk(ing) mode, if the FIFO becomes full and measurements are still being made, the new readings are discarded. In OVERwrite Mode, if the FIFO becomes full and measurements are still being made, new readings overwrite the oldest readings. In this situation, the FIFO contains only the latest 65,024 readings. In both modes Error +3021, "FIFO Overflow" is generated to let you know that measurements have been lost.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **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

```
SENS:DATA:FIFO:MODE OVERWRITE    Set FIFO to overwrite mode.
SENS: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 returned is **string**.
- **Related Commands:** SENSe:DATA:FIFO:MODE

Usage

```
SENSe: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 SENSe: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 returned 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 and Response Data" on page 156 of this chapter. For REAL,32, each reading is 4 bytes in length (the C-SCPI data type returned is **float32 array**). For REAL,64 and PACK,64, each reading is 8 bytes in length (the C-SCPI data type returned is **float64 array**).

Note Channel readings which are a positive overvoltage return `IEEE +INF` and a negative overvoltage return `IEEE -INF` (see the table on page 200 for actual values for each data format).

- **Related Commands:** SENSe:DATA:FIFO:COUNt?
- ***RST Condition:** FIFO buffer is empty.

Usage

```
SENSe: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

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** [SENSe:]DATA:FIFO:… commands.
- ***RST Condition:** SENSe:DATA:FIFO:RESet

Usage

```
SENS:DATA:FIFO:RES                    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.
- Autorange (the default voltage range) is not allowed while the A/D low-pass filter is enabled. If channels contained in the current Scan List specify autorange, while the A/D filter is ON an error will be generated at INITiate:IMMediate time.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- When the A/D filter is ON, the minimum allowable value for SAMPle:TIMer is 145 µsec.

Note If SAMPle:TIMer for the current scan list is less than 145 µsec and the A/D filter is ON, an error will be generated at INIT time.

- **Related Commands:** [SENSe:]FILTer[:LPASs][:STATe]?
- ***RST Condition:** SENSe:FILTer:LPASs:STATe OFF

Usage

```
SENS:FILT:LPAS:STAT 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

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Returned Value:** Returns numeric value `0` (off) or `1` (on). The C-SCPI type returned is **int16**.
- **Related Commands:** [SENSe:]FILTer[:LPASs][:STATe]

Usage

```
SENS:FILT:LPAS:STAT?          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 DIAGnostic:CUSTom:LINear or DIAGnostic:CUSTom:PIECewise 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)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 causes Error -222, "Data out of range" to occur. Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange.
- If you are using amplifier SCPs, you should set them first and keep their settings in mind when specifying a range setting. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.
- If an A/D reading is greater than the <*table_range*> specified with DIAGnostic:CUSTom:PIECewise, an overrange condition will occur.

- If no custom table has been loaded for the channels specified with SENSe:FUNCtion:CUSTom, Error +3056, "Custom EU not loaded" will be generated when an INITiate:IMMediate command is given.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- Related Commands: DIAGnostic:CUSTom:LINear, DIAGnostic:CUSTom:PIECewise
- ***RST Condition:** All custom EU tables are 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 channels 16 - 23.
SENS:FUNC:CUST  1,(@116:123)              Link custom EU with channels 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 DIAGnostic:CUSTom:PIECewise command. Measurements from a channel linked with SENSe:FUNCtion:CUSTom:REFerence will result in a temperature that is sent to the FIFO and/or CVT, as well as the module's Reference Temperature Register. This command is used to measure the temperature of an isothermal reference panel using custom characterized RTDs or thermistors. Contact your Hewlett-Packard System Engineer for more information on Custom Engineering Unit Conversion for your application.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*range*>	numeric (float32)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- See "Linking Channels to EU Conversion" in Chapter 3 for more information.
- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at

INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.
- The *CAL? command calibrates temperature channels based on Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- **Related Commands:** DIAGnostic:CUSTom:PIECewise, [SENSe:]FUNCtion:TEMPerature, [SENSe:]FUNCtion:CUSTom:TCouple, *CAL?
- ***RST Condition:** All custom EU tables are 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 reference temperature EU with channel 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 DIAGnostic:CUSTom:PIECewise 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. SENSe:FUNCtion:CUSTom:TCouple 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 | J | K | N | R | S | T	none
<*range*>	numeric (float32)	.0625 | .25 | 1 | 4 | 16 (See Comments)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- See "Linking Channels to EU Conversion" in Chapter 3 for more information.

- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.

- The <*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:** DIAGnostic:CUSTom:PIECewise, *CAL?, [SENSe:]REFerence, [SENSe:]REFerence:TEMPerature

- ***RST Condition:** All custom EU tables are 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 channels 0 - 7; use reference temperature compensation for N type wire.
SENS: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)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.

- Resistance measurements require the use of Current Source Signal Conditioning Plug-ons.

- The <*excite_current*> parameter (excitation current) does <u>not</u> 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:AMPLitude command. The choices for <*excite_current*> are 30E-6 (or MIN) and 488E-6 (or MAX). <*excite_current*> may be specified in milliamps (ma) and microamps (ua).

- The *CAL? command calibrates resistance channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.

- See "Linking Channels to EU Conversion" in Chapter 3 for more information.

- **Related Commands:** OUTPut:CURRent:AMPLitude, *CAL?

- ***RST Condition:** SENSe:FUNCtion:VOLTage AUTO,(@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 autorange (default).

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

Note on Syntax: Although the strain function is comprised of six separate SCPI commands, the only difference between them is the bridge type they specify to the strain EU conversion algorithm.

[SENSe:]FUNCtion:STRain:<*bridge_type*> [<*range*>,](@<*ch_list*>) links the strain EU conversion with the channels specified by <*ch_list*> to measure the bridge voltage. See "Linking Channels to EU Conversion" 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 (float32)	.0625 | .25 | 1 | 4 | 16 (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: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.
- The channel calibration command (*CAL?) calibrates the excitation voltage source on each Bridge Completion SCP.
- **Related Commands:** *CAL?, [SENSe:]STRain:… commands
- ***RST Condition:** SENSe:FUNCtion:VOLTage AUTO,(@100:163)

Usage

```
SENS:FUNC:STRain 1,(@100:,105,107)
```

[SENSe:]FUNCtion:TEMPerature

[SENSe:]FUNCtion:TEMPerature *<sensor_type>,<sub_type>*,[*<range>*,](@*<ch_list>*) links channels to an EU conversion for temperature based on the sensor specified in <*sensor_type*> and <*sub_type*>. **Not for sensing thermocouple reference temperature (for that, use the SENSe:REFerence command)**.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*sensor_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)	.0625 | .25 | 1 | 4 | 16 (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 to EU Conversion" in Chapter 3 for more information.

- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.

- The <*sub_type*> parameter: values of 85 and 92 differentiate between 100 Ohm (@ 0°C) RTDs with temperature coefficients of 0.00385 and and 0.00392 Ohm/Ohm/°C respectively. The <*sub_type*> values of 2250, 5000, and 10000 refer to thermistors that match the Omega 44000 series temperature response curve. These 44000 series thermistors are selected to match the curve within 0.1 or 0.2 °C. For thermistors <*sub_type*> may be specified in Kohms (kohm).

  The <*sub_type*> EEXTended applies to E type thermocouples at 800°C and above.

  CUSTom is pre-defined as Type K, with no reference junction compensation (reference junction assumed to be at 0°C).

- The *CAL? command calibrates temperature channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.

- **Related Commands:** *CAL?, OUTPut:CURRent (for RTDs and Thermistors), SENSe:REFerence, and SENSe:REFerence:TEMPerature (for Thermocouples)

- ***RST Condition:** [SENSe:]FUNCtion:VOLTage AUTO,(@100:163)

Usage

Link first 20 channels to the K type thermocouple temperature conversion.

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

Link channel 20 to measure reference temperature using 5K thermistor.

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

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

```
ROUT:SEQ:DEF  LIST1,(@120,100:119)
```

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

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

[SENSe:]FUNCtion:VOLTage[:DC]

[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)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.

- The *CAL? command calibrates channels based on Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.

- See "Linking Channels to EU Conversion" in Chapter 3 for more information.

- **Related Commands:** *CAL?, INPut:GAIN:… commands

- ***RST Condition:** SENSe:FUNCtion:VOLTage AUTO,(@100:163)

Usage

```
SENSe:FUNC:VOLT (@140:163)
```

Channels 40 - 63 measure voltage in autorange.

[SENSe:]REFerence

[SENSe:]REFerence <*sensor_type*>,<*sub_type*>,[<*range*>,](@<*ch_list*>) links the channel in <*ch_list*> to the reference junction temperature EU conversion based on <*sensor_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:REFerence:TEMPerature command. If used, the reference junction channel must be scanned before any thermocouple channels.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*sensor_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)	.0625 | .25 | 1 | 4 | 16 (See Comments below)	VDC
<*ch_list*>	channel list (string)	100 - 163	none

Comments

- See "Linking Channels to EU Conversion" in Chapter 3 for more information.
- The <*range*> parameter: The HP E1413 has five ranges: .0625 VDC, .25 VDC, 1 VDC, 4 VDC, and 16 VDC. To select a range, simply specify the range value (for example, 4 selects the 4 VDC 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 16 VDC range). Specifying a value larger than 16 generates Error -222, "Data out of range". Specifying 0 selects the lowest range (.0625 VDC). Specifying AUTO selects autorange. The default range (no range parameter specified) is autorange. Autorange is not allowed while SENSe:FILTer:LPASs:STATe is ON. Error +3072, "Autorange not allowed with SENSE:FILTER on" will be generated at INITiate:IMMediate time if the filter is ON and any channel specifies autorange. <*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. For instance, if your expected signal voltage is to be approximately .1 VDC and the amplifier SCP for that channel has a gain of 8, you must set <*range*> no lower than 1 VDC or an input out-of-range condition will exist.

- The <*sensor_type*> parameter specifies the sensor type that will be used to determine the temperature of the isothermal reference panel. <*sensor_type*> CUSTom is pre-defined as Type E with 0°C reference junction temperature, and is not re-defineable. The user is responsible for providing an E type thermocouple with 0°C reference junction.
- For <*sensor_type*> THERmistor, the <*sub_type*> parameter may be specified in ohms or kohm.
- The *CAL? command calibrates resistance channels based on Current Source SCP and Sense Amplifier SCP setup at the time of execution. If SCP settings are changed, those channels are no longer calibrated. *CAL? must be executed again.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** [SENSe:]FUNCtion:TEMPerature, [SENSe:]FUNCtion:CUSTom:REFerence
- ***RST Condition:** Reference temperature is 0 °C

Usage *Sense the reference temperature on channel 20 using an RTD.*

```
SENS: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:REFerence:TEMPerature value or by scanning a channel linked with the SENSe:REFerence command. If used, SENSe:REFerence:TEMPerature 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.
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.

- **Related Commands:** [SENSe:]FUNCtion:TEMPerature
- ***RST Condition:** Reference temperature is 0 °C

Usage

```
SENS: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, but declares the excitation voltage applied by a voltage source outside the HP E1413.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*excite_v*>	numeric (float32)	.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:… commands, [SENSe:]FUNCtion:STRain:… commands
- ***RST Condition:** 3.9 V

Usage

```
SENS:STRAIN:EXC  4,(@100:107)
```

Set excitation voltage for channels 0 through 7 to +4 V.

[SENSe:]STRain:EXCitation?

[SENSe:]STRain:EXCitation? (@*<channel>*) returns the excitation voltage value currently set for the sense channel specified by <*channel*>.

Parameters

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

Comments

- **Returned Value:** Numeric value of excitation voltage. The C-SCPI type returned is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** [SENSe:]STRain:EXCitation

Usage SENS:STRAIN:EXC? (@107) *Query excitation voltage for channel 7.*

Enter statement here. *Returns the excitation voltage set by STR:EXC.*

[SENSe:]STRain:GFACtor

[SENSe:]STRain:GFACtor <*gage_factor*>,(@<*ch_list*>) specifies the gage factor to be used to convert strain bridge readings for the channels specified by <*ch_list*>.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*gage_factor*>	numeric (float32)	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:GFACtor?, [SENSe:]FUNCtion:STRain:… commands
- ***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

- <*channel*> must specify a single channel only.
- **Returned Value:** Numeric value of gage factor. The C-SCPI type returned is **flt32**.
- **Related Commands:** [SENSe:]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 (float32)	.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:** [SENSe:]FUNCtion:STRain:… commands, [SENSe:]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*>	channel list (string)	100 - 163	none

Comments

- **Returned Value:** Numeric value of the Poisson ratio. C-SCPI type returned is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** [SENSe]FUNCtion:STRain:… commands, [SENSe:]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 (float32)	-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:UNSTrained?, [SENSe:]FUNCtion:STRain:… commands
- ***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 C-SCPI type returned is **flt32**.
- <*channel*> must specify a single channel only.
- **Related Commands:** [SENSe:]STRain:UNSTrained

Usage

```
STRAIN:UNST? (@107)            Query unstrained voltage for channel 7.
Enter statement here.          Returns the unstrained voltage set by
                               STR:UNST.
```

STATus

The STATus subsystem communicates with the SCPI defined Operation and Questionable Data Status Register sets. Each is comprised of a Condition Register, a set of Positive and Negative Transition Filter Registers, an Event Register, and an Enable Register. Condition Registers allow you to view the current real-time states of their status signal inputs (signal states are not latched). The Positive and Negative Transition Filter Registers allow you to control the polarity of change from the Condition Registers that will set Event Register bits. Event Registers contain latched representations of signal transition events from their Condition Register. 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.

Subsystem Syntax

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

The status system contains four status groups.

- Operation Status Group
- Questionable Data Group
- Standard Event Group
- Status Byte Group

This SCPI STATus subsystem communicates with the first two groups while IEEE-488.2 Common Commands (documented later in this chapter) communicate with Standard Event and Status Byte Groups.

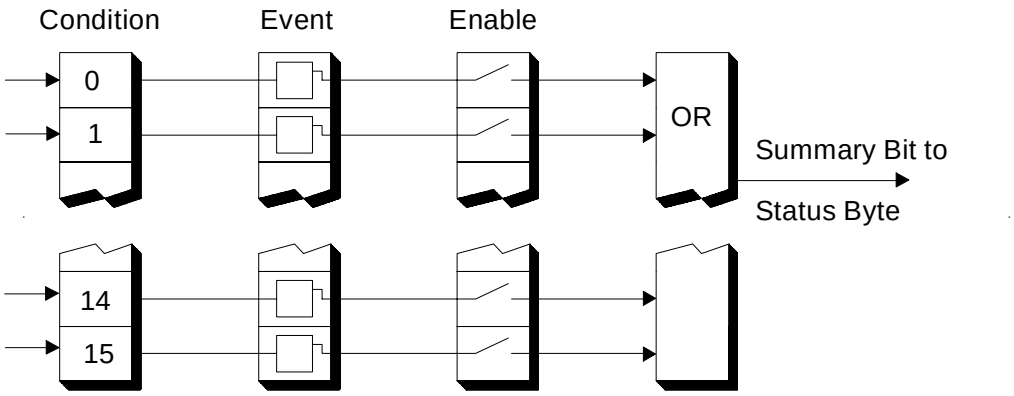


Figure 5-4. General Status Register Organization

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. See Table 5-3 for the bit assignments.

Table 5-3. Operation Status Group

Bit #	Dec Value	Hex Value	Bit Name	Description
0	1	0001_{16}	Calibrating	Set by CALibration:TARE, and CALibration:SETup. Cleared by CALibration:TARE?, and CALibration:SETup?. Set while *CAL? executes and reset when *CAL? completes. Set by CALibration:CONFigure:VOLTage or CALibration:CONFigure:RESistance. Cleared by CALibration:VALue:VOLTage or CALibration:VALue:RESistance.
1-3			Not Used	
4	16	0010_{16}	Measuring	Set when instrument INITiated. Cleared when instrument returns to Trigger Idle State.
5-7			Not Used	
8	256	0100_{16}	Scan Complete	Set when each pass through a Scan List completed (may not indicate all measurements have been taken when INITiate:CONTinuous is ON or TRIGger:COUNt >1).
9	512	0200_{16}	SCP Trigger	An SCP has sourced a trigger event (future HP 1413 SCPs).
10	1024	0400_{16}	FIFO Half Full	The FIFO contains at least 32,768 readings.
11	2048	0800_{16}	Limit Test Exceeded	One or more limit tests was exceeded.
12-15			Not Used	

STATus:OPERation:CONDition?

STATus:OPERation:CONDition? returns the decimal weighted value of the bits set in the Condition Register.

Comments

- The Condition Register reflects the real-time state of the status signals. The signals are not latched, and therefore, past events are not retained in this register (see STATus:OPERation[:EVENt]?).
- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** *CAL?, CALibration:ZERO, INITiate[:IMMediate], STATus:OPERation[:EVENt]?, STATus:OPERation:ENABle, STATus:OPERation:ENABle?
- ***RST Condition:** No Change.

Usage

```
STATUS:OPERATION:CONDITION?
```

Enter statement will return value from the Condition Register.

STATus:OPERation:ENABle

STATus:OPERation:ENABle ***<enable_mask>*** sets bits in the Enable Register that will enable corresponding bits from the Event Register to set the Operation Summary bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*enable_mask*>	numeric (uint16)	0 - 32767	none

Comments

- <*enable_mask*> may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- **VXI Interrupts:** When Operation Status Group bits 4, 8, 9, 10, or 11 are enabled, VXI card interrupts will occur as follows:

  When the event corresponding to bit 4 occurs and then is cleared, the card will generate a VXI interrupt. When the event corresponding to bit 8, 9, 10, or 11 occurs, the card will generate a VXI interrupt.

  NOTE: In C-SCPI, the C-SCPI overlap mode must be on for VXIbus interrupts to occur.
- **Related Commands:** *STB?, SPOLL, STATus:OPERation:CONDition?, STATus:OPERation[:EVENt]?, STATus:OPERation:ENABle?
- **Cleared By:** STATus: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 returned is **uint16**.
- **Related Commands:** *STB?, SPOLL, STATus:OPERation:CONDition?, STATus:OPERation[:EVENt]?, STATus:OPERation: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 returned is **uint16**.
- **Related Commands:** *STB?, SPOLL, STATus:OPERation:CONDition?, STATus:OPERation:ENABle, STATus:OPERation:ENABle?
- **Cleared By:** *CLS, power-on, and by reading the register.
- ***RST Condition:** No change.

Usage

STAT:OPER:EVENT?	*Enter statement will return the value of bits set in the Operation Event Register.*
STAT:OPER?	*Same as above.*

STATus:OPERation:NTRansition

STATus:OPERation:NTRansition *<transition_mask>* sets bits in the Negative Transition Filter (NTF) Register. When a bit in the NTF Register is set to one, the corresponding bit in the Condition Register must change from a one to a zero in order to set the corresponding bit in the Event Register. When a bit in the NTF Register is zero, a negative transition of the Condition Register bit will not change the Event Register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*transition_mask*>	numeric (uint16)	0 - 32767	none

Comments

- <*transition_mask*> may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the Status Operation Positive Transition Filter (STAT:OPER:PTR) and Status Operation Negative Transition Filter (STAT:OPER:NTR) Registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event Register.
- If neither the Status Operation Positive Transition Filter (STAT:OPER:PTR) or Status Operation Negative Transition Filter (STAT:OPER:NTR) Registers have a corresponding bit set to one, transitions from the Condition Register will have no effect on the Event Register.

- **Related Commands:** STATus:OPERation:NTRansition?, STATus:OPERation:PTRansition
- **Cleared By:** STATus:PRESet and power-on.
- ***RST Condition:** No change.

Usage

```
STAT:OPER:NTR  16
```

When "Measuring" bit goes false, set bit 4 in Status Operation Event Register.

STATus:OPERation:NTRansition?

STATus:OPERation:NTRansition? returns the value of bits set in the Negative Transition Filter (NTF) Register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** STATus:OPERation:NTRansition
- ***RST Condition:** No change.

Usage

```
STAT:OPER:NTR?
```

Enter statement returns current value of bits set in the NTF register.

STATus:OPERation:PTRansition

STATus:OPERation:PTRansition *<transition_mask>* sets bits in the Positive Transition Filter (PTF) Register. When a bit in the PTF Register is set to one, the corresponding bit in the Condition Register must change from a zero to a one in order to set the corresponding bit in the Event Register. When a bit in the PTF Register is zero, a positive transition of the Condition Register bit will not change the Event Register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*transition_mask*>	numeric (uint16)	0 - 32767	none

Comments

- <*transition_mask*> may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the Status Operation Positive Transition Filter (STAT:OPER:PTR) and Status Operation Negative Transition Filter (STAT:OPER:NTR) Registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event Register.
- If neither the Status Operation Positive Transition Filter (STAT:OPER:PTR) or Status Operation Negative Transition Filter (STAT:OPER:NTR) Registers have a

corresponding bit set to one, transitions from the Condition Register will have no effect on the Event Register.

- **Related Commands:** STATus:OPERation:NTRansition, STATus:OPERation:PTRansition?
- **Set to all ones by:** STATus:PRESet and power-on.
- ***RST Condition:** No change.

Usage

```
STAT:OPER:PTR  16
```

When "Measuring" bit goes true, set bit 4 in Status Operation Event Register.

STATus:OPERation:PTRansition?

STATus:OPERation:PTRansition? returns the value of bits set in the Positive Transition Filter (PTF) Register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** STATus:OPERation:PTRansition
- ***RST Condition:** No change.

Usage

```
STAT:OPER:PTR?
```

Enter statement returns current value of bits set in the PTF Register.

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, STATus:OPERation:ENABle, STATus:OPERation:ENABle?, STATus:QUEStionable:ENABle, STATus:QUEStionable:ENABle?
- ***RST Condition:** No change.

Usage

```
STAT:PRESET
```

Clear both of the Enable Registers.

Questionable Data Group

The Questionable Data Group indicates when errors are causing lost or questionable data. See Table 5-4 for the bit assignments.

Table 5-4. Questionable Data Group

Bit #	Dec Value	Hex Value	Bit Name	Description
0 - 7	Not Used			
8	256	0100_{16}	Calibration Lost	At *RST or Power-on Control Processor has found a checksum error in the Calibration Constants. Read error(s) with SYSTem:ERRor? and re-calibrate area(s) that lost constants.
9	512	0200_{16}	Trigger Too Fast	Scan not complete when another trigger event received.
10	1024	0400_{16}	FIFO Overflowed	Attempt to store more than 65,024 readings in FIFO.
11	2048	0800_{16}	Over voltage Detected on Input	If the input protection jumper has not been cut, the input relays have been opened and *RST is required to reset the module. Overvoltage will also generate an error.
12	4096	1000_{16}	VME Memory Overflow	The number of readings taken exceeds VME memory space.
13	8192	2000_{16}	Setup Changed	Channel Calibration in doubt because SCP setup may have changed since last *CAL? or CALibration: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, and therefore, past events are not retained in this register (see STATus:QUEStionable[:EVENt]?).
- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** CALibration:VALue:RESistance, CALibration:VALue:VOLTage, STATus:QUEStionable[:EVENt]?, STATus:QUEStionable:ENABle, STATus:QUEStionable:ENABle?
- ***RST Condition:** No change.

Usage

```
STATUS:QUESTIONABLE:CONDITION?
```

Enter statement will return value from the 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, STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABle?, STATus:QUEStionable[:EVENt]?
- **Cleared By:** STATus: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 returned is **uint16**.
- **Related Commands:** *STB?, SPOLL, STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABle, STATus:QUEStionable[:EVENt]?
- ***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 returned is **uint16**.
- **Cleared By:** *CLS, power-on, and by reading the register.
- **Related Commands:** *STB?, SPOLL, STATus:QUEStionable:CONDition?, STATus:QUEStionable:ENABle, STATus:QUEStionable:ENABle?

Usage

STAT:QUES:EVENT?	*Enter statement will return the value of bits set in the Questionable Event Register.*
STAT:QUES?	*Same as above.*

STATus:QUEStionable:NTRansition

STATus:QUEStionable:NTRansition *<transition_mask>* sets bits in the Negative Transition Filter (NTF) Register. When a bit in the NTF Register is set to one, the corresponding bit in the Condition Register must change from a one to a zero in order to set the corresponding bit in the Event Register. When a bit in the NTF Register is zero, a negative transition of the Condition Register bit will not change the Event Register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*transition_mask*>	numeric (uint16)	0 - 32767	none

Comments

- <*transition_mask*> may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the Status Questionable Positive Transition Filter (STAT:QUES:PTR) and Status Questionable Negative Transition Filter (STAT:QUES:NTR) Registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event Register.
- If neither the Status Questionable Positive Transition Filter (STAT:QUES:PTR) or Status Questionable Negative Transition Filter (STAT:QUES:NTR) Registers have a corresponding bit set to one, transitions from the Condition Register will have no effect on the Event Register.
- **Related Commands:** STATus:QUEStionable:NTRansition?, STATus:QUEStionable:PTRansition
- **Cleared By:** STATus:PRESet and power-on.

- ***RST Condition:** No change.

Usage

```
STAT:QUES:NTR  1024
```

When "FIFO Overflowed" bit goes false, set bit 10 in Status Questionable Event Register.

STATus:QUEStionable:NTRansition?

STATus:QUEStionable:NTRansition? returns the value of bits set in the Negative Transition Filter (NTF) Register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** STATus:QUEStionable:NTRansition
- ***RST Condition:** No change.

Usage

```
STAT:QUES:NTR?
```

Enter statement returns current value of bits set in the NTF Register.

STATus:QUEStionable:PTRansition

STATus:QUEStionable:PTRansition *<transition_mask>* sets bits in the Positive Transition Filter (PTF) Register. When a bit in the PTF Register is set to one, the corresponding bit in the Condition Register must change from a zero to a one in order to set the corresponding bit in the Event Register. When a bit in the PTF register is zero, a positive transition of the Condition Register bit will not change the Event Register bit.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*transition_mask*>	numeric (uint16)	0 - 32767	none

Comments

- <*transition_mask*> may be sent as decimal, hex (#H), octal (#Q), or binary (#B).
- If both the Status Questionable Positive Transition Filter (STAT:QUES:PTR) and Status Questionable Negative Transition Filter (STAT:QUES:NTR) Registers have a corresponding bit set to one, any transition, positive or negative will set the corresponding bit in the Event Register.
- If neither the Status Questionable Positive Transition Filter (STAT:QUES:PTR) or Status Questionable Negative Transition Filter (STAT:QUES:NTR) Registers have a corresponding bit set to one, transitions from the Condition Register will have no effect on the Event Register.
- **Related Commands:** STATus:QUEStionable:NTRansition, STATus:QUEStionable:PTRansition?

- **Set to all ones by:** STATus:PRESet and power-on.
- ***RST Condition:** No change.

Usage

```
STAT:QUES:PTR  1024
```

When "FIFO Overflowed" bit goes true, set bit 10 in Status Operation Event Register.

STATus:QUEStionable:PTRansition?

STATus:QUEStionable:PTRansition? returns the value of bits set in the Positive Transition Filter (PTF) Register.

Comments

- **Returned Value:** Decimal weighted sum of all set bits. The C-SCPI type returned is **uint16**.
- **Related Commands:** STATus:QUEStionable:PTRansition
- ***RST Condition:** No change.

Usage

```
STAT:OPER:PTR?
```

Enter statement returns current value of bits set in the PTF Register.

SYSTem

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 Model <*model number and description*> SCP,0,0
- The C-SCPI type returned 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:CTYP? (@100)          Return SCP type install at channel 0.
```

SYSTem:ERRor?

SYSTem:ERRor? returns the latest error entered into the Error Queue.

Comments

- SYSTem:ERRor? returns one error message from the Error Queue (returned error is removed from queue). To return all errors in the queue, repeatedly execute SYSTem:ERRor? 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 returned is **string**.

Usage SYST:VER? *Returns "1990".*

TRIGger

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.

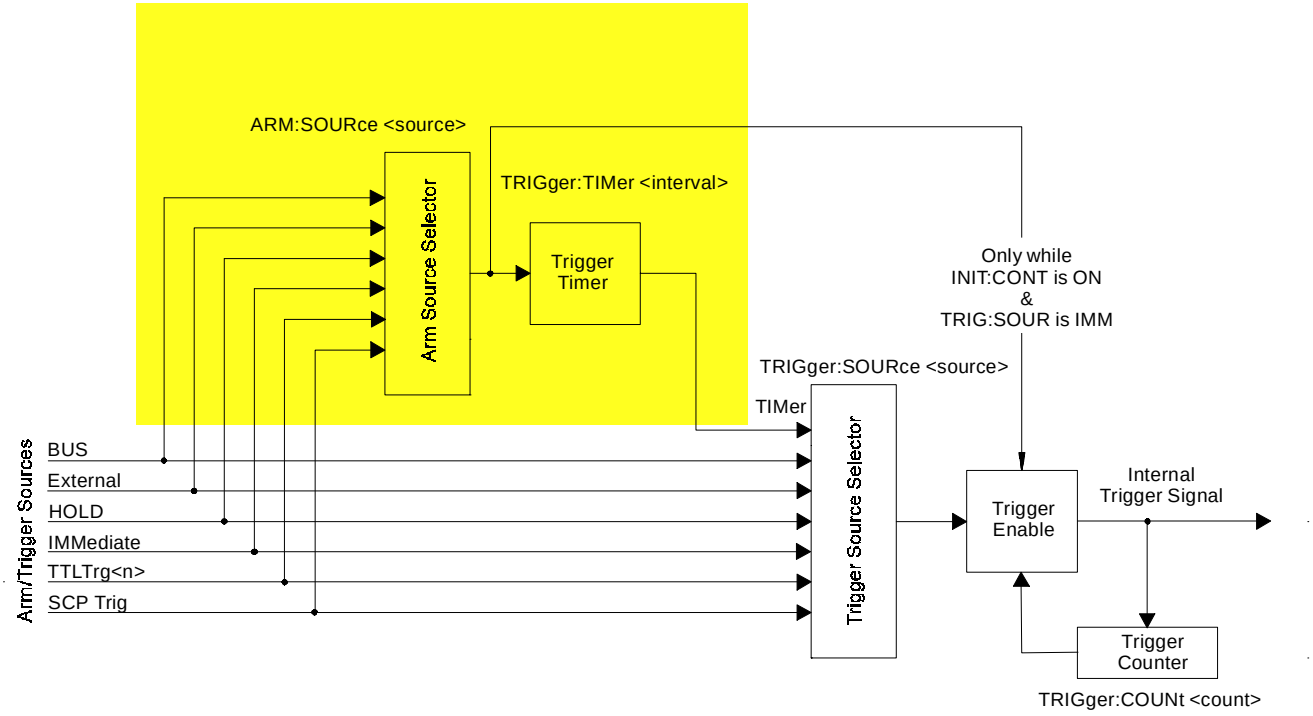


Figure 5-5. Logical Trigger Model

Note The ARM system only exists while TRIGger:SOURce is TIMer or while the Continuous Mode is set (TRIGger:SOURce IMMediate and INITiate:CONTinuous ON). When neither of the above conditions are true, SCPI compatibility requires that ARM:SOURce be set to IMMediate or Error -221, "Settings conflict" will be generated.

Event Sequence Figure 5-6 shows how the module responds to various trigger/arm configurations.

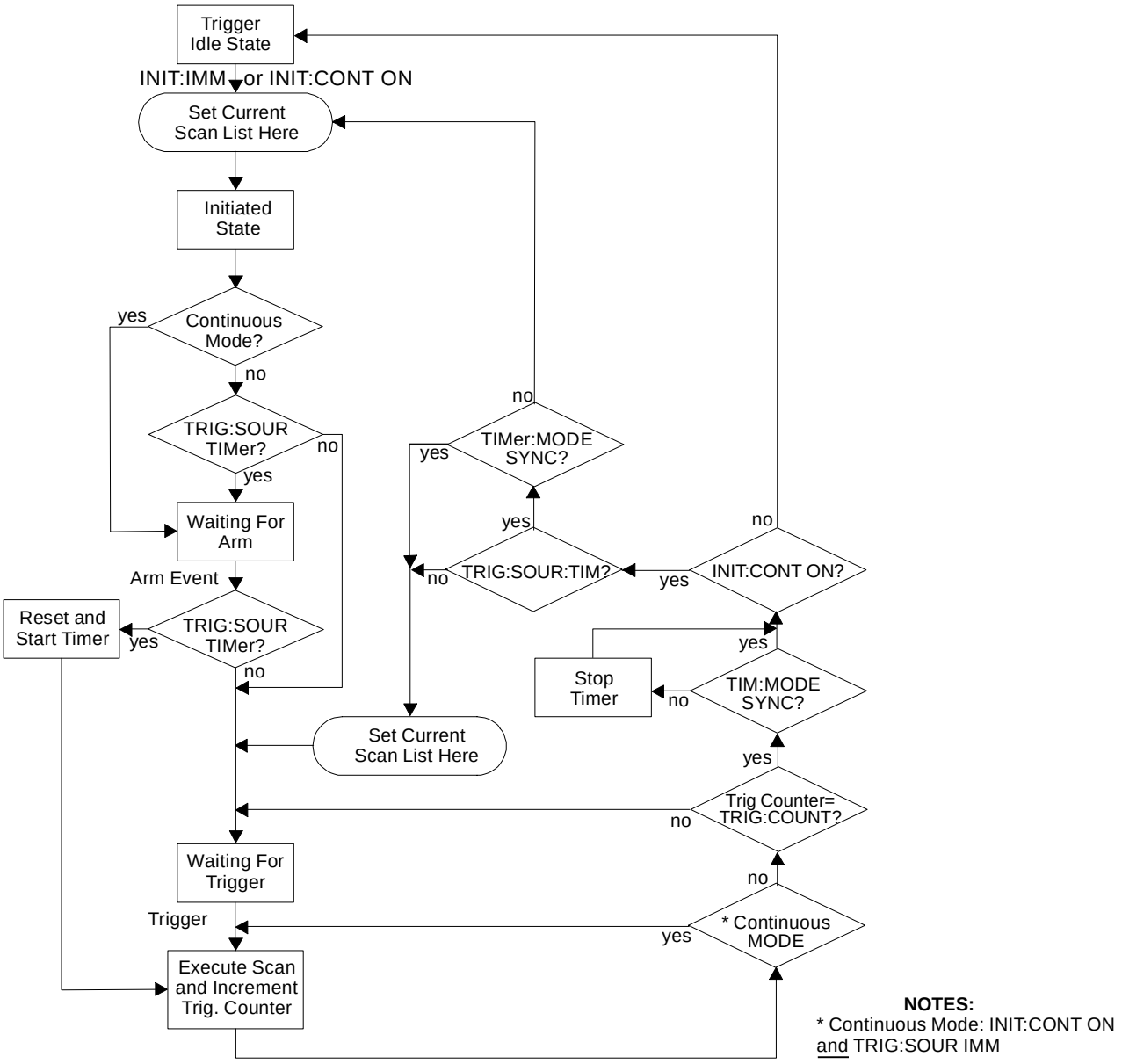


Figure 5-6. Trigger/Scan Sequence Diagram

Subsystem Syntax

```
TRIGger
    :COUNt <trig_count>
    :COUNt?
    [:IMMediate]
    :SOURce <source>
    :SOURce?
    :TIMer
        :MODE <mode>
        :MODE?
        [:PERiod] <trig_interval>
        [:PERiod]?
```

TRIGger:COUNt

TRIGger:COUNt *<trig_count>* sets the number of times the module can be triggered before it returns to the Trigger Idle State. The default count is 1 (returns to Idle State after each trigger). See Figure 5-6.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*trig_count*>	numeric (uint16) (string)	0 to 65535 | INF	none

Comments

- When <*trig_count*> is set to 0 or INF, the trigger counter is disabled. Once INITiated the module will return to the Wait-for-Trigger State after each trigger event. The ABORt (preferred) and *RST commands will return the module to the Trigger Idle State. ABORt is preferred since *RST also returns other module configurations to their default settings.
- The module will not change Scan Lists until the trigger count is completed. See Figure 5-3 on page 221.
- The default count is 1.
- **Related Commands:** TRIGger:COUNt?, ROUTe:SCAN LIST1-LIST4
- ***RST Condition:** TRIGger:COUNt 1

Usage

TRIG:COUN 10	*Set the module to make 10 passes through a Scan List.*
TRIG:COUN 0	*Set the module to accept unlimited triggers.*

TRIGger:COUNt?

TRIGger:COUNt? returns the currently set trigger count.

Comments

- If TRIGger: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 returned is **int32**.
- **Related Commands:** TRIGger:COUNt
- ***RST Condition:** TRIGger:COUNt? returns 1.

Usage

TRIG:COUN?	*Query for trigger count setting.*
Enter statement.	*Returns the TRIG:COUN setting.*

TRIGger[:IMMediate]

TRIGger[:IMMediate] causes one trigger when the module is set to the TRIGger:SOURce BUS or TRIGger:SOURce HOLD mode.

Comments

- This command is equivalent to the *TRG common command or the IEEE-488.2 GET bus command.
- **Related Commands:** TRIGger:SOURce

Usage TRIG:IMM *Use TRIGGER to start a measurement scan.*

TRIGger:SOURce

TRIGger:SOURce *<trig_source>* configures the trigger system to respond to the trigger event.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*trig_source*>	discrete (string)	BUS | EXT | HOLD | IMM | SCP | TIM | TTLTrg<*n*>	none

HP E1313 Note Most B-Size mainframes do not support VXIbus TTLTRG lines.

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 event is always satisfied.
SCP	SCP Trigger Bus (future HP or SCP Breadboard)
TIMer	The internal trigger timer. See also TRIGger:TIMer:MODE
TTLTrg<*n*>	The VXIbus TTLTRG lines (n=0 through 7)

- See the note about the ARM subsystem on page 266.
- The term "Continuous Mode" means that the module is scanning with TRIGger:SOURce IMMediate and INITiate:CONTinuous ON. After an ARM event, the module executes a single Scan List continuously until an INITiate:CONTinuous OFF is received. Continuous Mode provides the fastest scan rate.

- The Trigger and Arm Sources can be changed while the module is scanning (but not in the Continuous Mode). This provides a way to stop triggering when you want to change the TRIGger:TIMer:PERiod <*interval*>. To do this, execute:

ARM:SOUR IMM	*Must be IMM when SOUR not TIMer.*
TRIG:SOUR BUS or HOLD	*To stop scanning.*
TRIG:TIM <*new_interval*>	*Change the timer interval.*
TRIG:SOUR TIM	*Return to timed scans.*

- See TRIGger:TIMer:MODE for scanning behavior while TRIGger:SOURce is TIMer.
- While TRIGger:SOURce is IMMediate, you need only INITiate the trigger system to start a measurement scan.
- **Related Commands:** ABORt, INITiate:… commands, *TRG
- ***RST Condition:** TRIGger:SOURce 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 returned is **string**. See the TRIGger:SOURce command for more response data information.

Usage

TRIG:SOUR?	*Ask HP E1413 to return trigger source configuration.*

TRIGger:TIMer:MODE

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

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*timer_mode*>	discrete (string)	SYNChronous | ASYNchronous	none

Comments

- To have time-triggered scans while TRIGger:TIMer:MODE is ASYN, you set TRIGger:COUNt to the number of scans you wish to execute (or INF for continuous scans). A Timer Arm event will immediately trigger the first scan and start the Timer. When the trigger count is reached and INITiate:CONTinuous is ON, the module stops the trigger timer and returns to the Waiting For Arm State. At this point, any asynchronous Arm event can immediately trigger a scan and

restart the timer. Use this mode when you want the Timer Arm to control the start of multiple timed scans. See Figure 5-7 for operation.

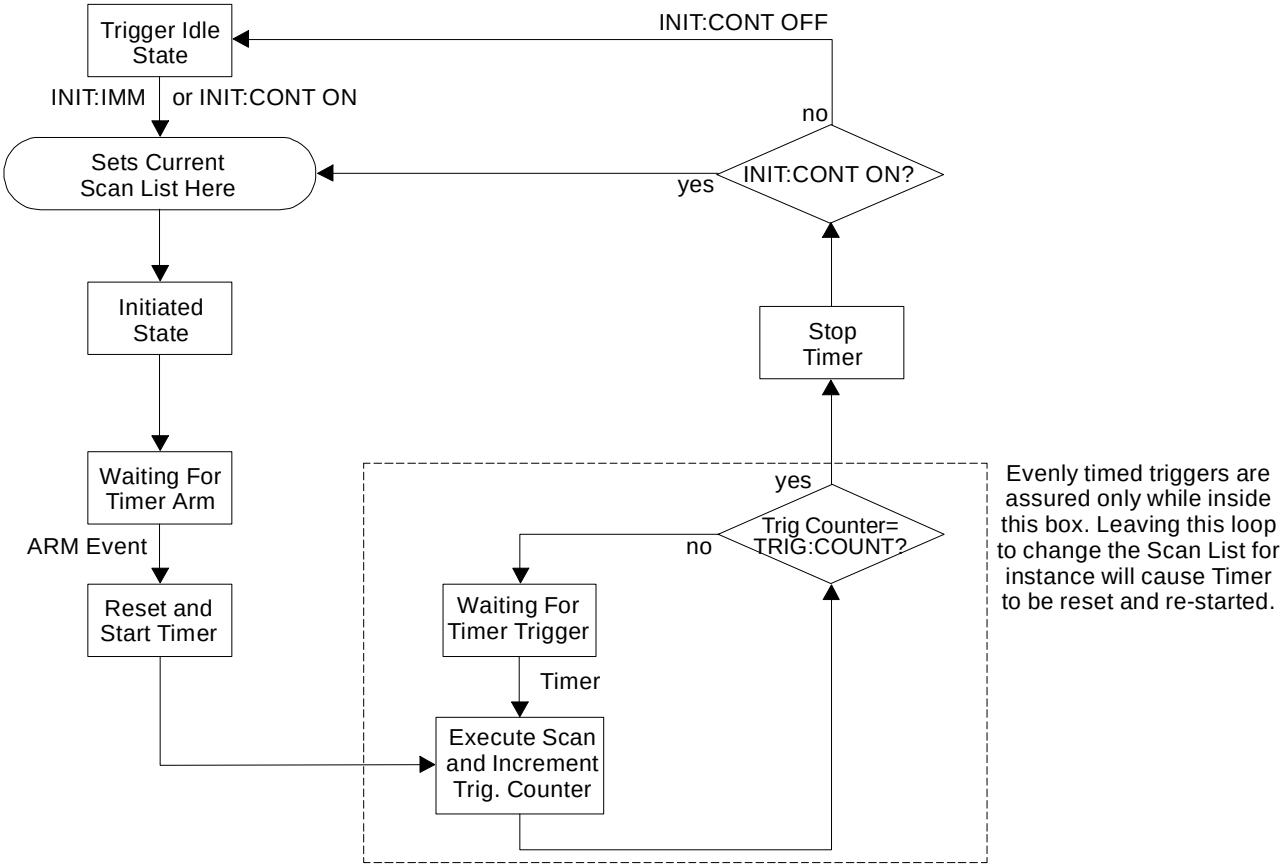


Figure 5-7. Asynchronous Mode Sequence

- Use TRIGger:TIMer:MODE SYNC with INITiate:CONTinuous ON when you want to be able to change the Scan List while performing synchronously timed scans. In SYNC mode a Timer Arm event immediately triggers a scan and starts the timer. When the trigger count is reached and INITiate:CONTinuous is ON, the trigger timer continues running, the current Scan List is read, and the module returns to wait for the next Timer trigger. See Figure 5-8 for operation.

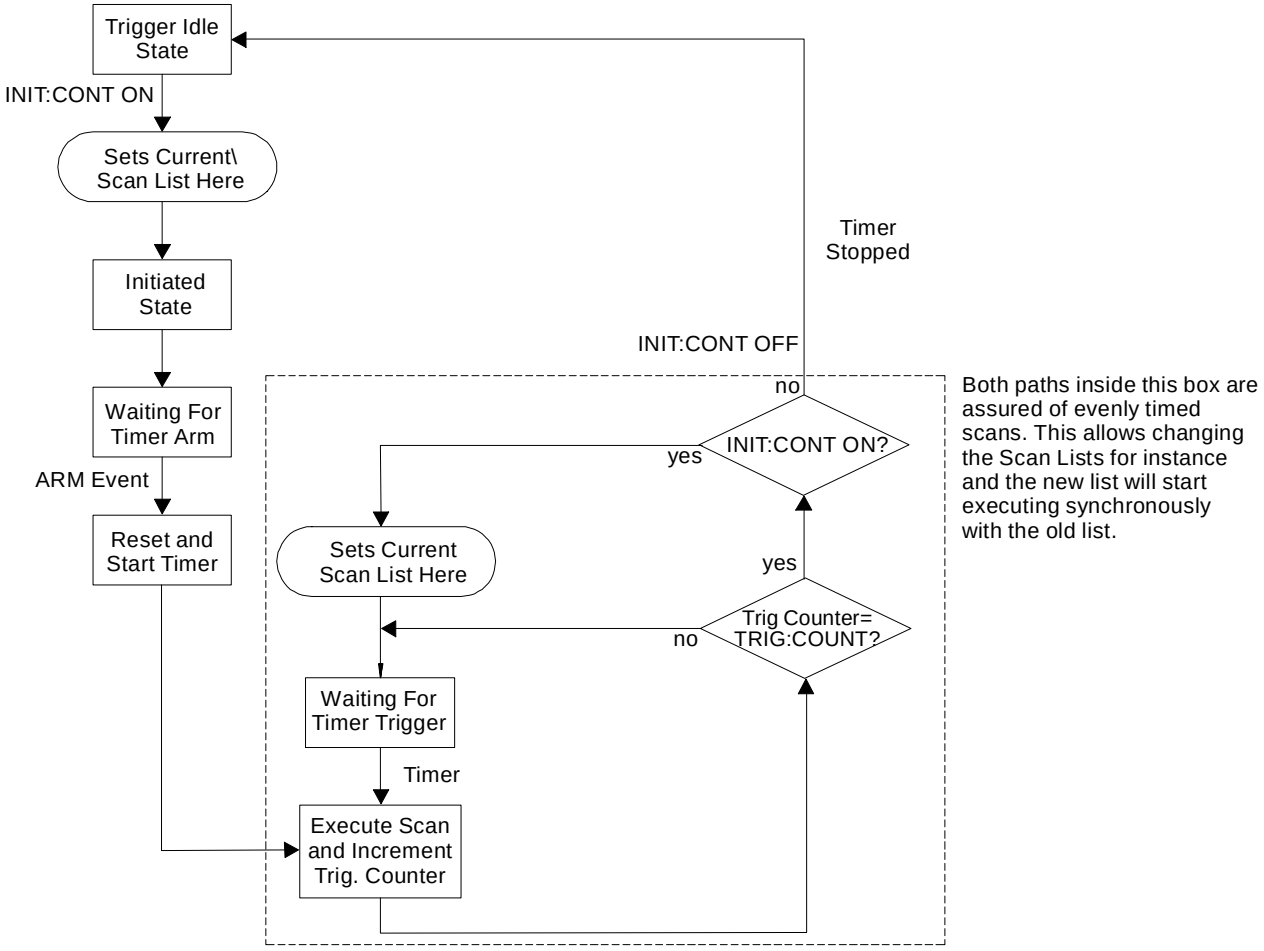


Figure 5-8. Synchronous Mode Sequence

- **Related Commands:** INITiate:CONTinuous, TRIGger:COUNt, TRIGger:TIMer:MODE?
- ***RST Condition:** TRIGger:TIMer:MODE ASYNCHRONOUS

Usage TRIG:TIM:MODE SYNC *Change trigger 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 returned is **string**.
- **Related Commands:** TRIGger:TIMer:MODE
- ***RST Condition:** TRIGger:TIMer:MODE ASYNCHRONOUS

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

TRIGger:TIMer[:PERiod]

TRIGger:TIMer[:PERiod] <*trig_interval*> sets the interval between scan triggers. Used with the TRIGger:SOURce TIMer trigger mode.

Parameters

Parameter Name	Parameter Type	Range of Values	Default Units
<*trig_interval*>	numeric (float32) (string)	1.0E-4 to 6.5536 | MIN | MAX	seconds

Comments

- In order for the TRIGger: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. <*trig_interval*> may be specified in milliseconds (ms), or microseconds (us). The resolution for <*trig_interval*> is 100 µs.

Notes

1. To change the TRIGger:TIMer:PERiod <*trig_interval*> without re-INITializing the module, execute:

ARM:SOUR IMM	*Must be IMM when SOUR not TIMer.*
TRIGger:SOURce BUS *or* HOLD	*To stop scanning.*
TRIG:TIM <*new_interval*>	*Change the timer interval.*
TRIG:SOUR TIM	*Return to timed scans.*

2. To avoid a "Trigger Too Fast" error following INITiate:IMMediate, the TRIGger:TIMer *interval* must be greater than:
   ((number of channels in scan list +3)*SAMPle:TIMer) +30µs.

- **Related Commands:** TRIGger:SOURce TIMer, ARM:SOURce, ARM[:IMMediate], INITiate:… commands, TRIGger:SOURce?
- ***RST Condition:** TRIGger:TIMer:PERiod 1.0E-3

Usage

TRIG:TIM:PER 1.0E-2	*Set the module to make a measurement scan every 10 mS.*
TRIG:TIM:PER 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 returned is **float32**.
- **Related Commands:** TRIGger:TIMer[:PERiod]
- ***RST Condition:** 1.0E-4

Usage

```
TRIG:TIM:PER?                    Query trigger timer.
Enter statement.                 Returns the timer setting.
```

Common Command Reference

The following reference discusses the HP E1313/E1413 IEEE-488.2 Common Commands.

*CAL?

Calibration Command. The calibration command causes the Channel Calibration function to be performed for every module channel. The Channel Calibration function includes calibration of A/D Offset, and Gain and Offset for all 64 channels. This calibration is accomplished using internal calibration references. The *CAL? command causes the module to calibrate A/D offset and gain, and all channel offsets. This may take many minutes to complete. The actual time it will take your HP E1313/E1413 to complete *CAL? depends on the mix of SCPs installed. *CAL? performs literally hundreds of measurements of the internal calibration sources for each channel and must allow 17 time constants of settling wait each time a filtered channel's calibrations source value is changed. The *CAL? procedure is internally very sophisticated and results in an extremely well calibrated module.

To perform Channel Calibration on multiple HP E1313/E1413s, use the CALibration:SETup command (see CALibration:SETup on page 179 for details).

- **Returned Value:**

Value	Meaning	Further Action
0	Cal OK	None
-1	Cal Error	Query the Error Queue (SYSTem:ERRor?) See Error Messages in Appendix B

The C-SCPI type returned is **int16**.

- CALibration:STORe ADC stores the calibration constants for *CAL? and CALibration:SETup into non-volatile memory.
- Executing this command **does not** alter the module's programmed state (function, range, etc.), but it does clear bit 13 in the Questionable Data Group (see Table 5-4 on page 259).
- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Related Commands:** CALibration:SETup, CALibration:SETup?, CALibration:STORe ADC

Note If Open Transducer Detect (OTD) is enabled when *CAL? is executed, the module will disable OTD, wait 1 minute to allow channels to settle, perform the calibration, and then re-enable OTD. If your program turns off OTD before executing *CAL?, it should also wait 1 minute for settling.

*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, and 7) in the Status Byte Register. *CLS does not affect the enable bits in any of the status register groups. (The SCPI command STATus:PRESet does clear the Operation Status Enable and Questionable Data Enable Registers.) *CLS disables the Operation Complete function (*OPC command) and the Operation Complete Query function (*OPC? command).

*DMC <*name*>,<*cmd_data*>

Define Macro Command. Assigns one, or a sequence of commands to a named macro.

The command sequence may be composed of SCPI and/or Common Commands. <*name*> may be the same as a SCPI command, but may not be the same as a common command. When a SCPI named macro is executed, the macro rather than the SCPI command is executed. To regain the function of the SCPI command, execute *EMC 0.

<*cmd_data*> is sent as *arbitrary block program data*. For more information see page 156.

*EMC

Enable Macro Command. When <*enable*> is non-zero, macros are enabled. When <*enable*> is zero, macros are disabled.

*EMC?

Enable Macro Query. Returns either 1 (macros are enabled), or 0 (macros are disabled).

*ESE <*mask*>

Standard Event Status Enable Register Command. Enables one or more events in the Standard Event Status Register to be reported in bit 5 (the Standard Event Status Summary bit) of the Status Byte Register. You enable an event by specifying its decimal weight for <*mask*>. To enable more than one event (bit), specify the sum of the decimal weights. The C-SCPI type for <*mask*> is **int16**.

Bit #	7	6	5	4	3	2	1	0
Weighted Value	128	64	32	16	8	4	2	1
Event	Power-on	User Request	Command Error	Execution Error	Device Dependent Error	Query Error	Request Control	Operation Complete

*ESE?

Standard Event Status Enable Query. Returns the weighted sum of all enabled (unmasked) bits in the Standard Event Status Register. The C-SCPI type for this returned value is **int16**.

*ESR?

Standard Event Status Register Query. Returns the weighted sum of all set bits in the Standard Event Status Register. After reading the register, *ESR? clears the register. The events recorded in the Standard Event Status Register are independent of whether or not those events are enabled with the *ESE command to set the Standard Event Summary bit in the Status Byte Register. The Standard Event bits are described in the *ESE command. The C-SCPI type for this returned value is **int16**.

*GMC? <*name*>

Get Macro Query. Returns arbitrary block response data which contains the command or command sequence defined for <*name*>. For more information on arbitrary block response data see page 156.

*IDN?

Identity Query. Returns the device identity. The response consists of the following four fields (fields are separated by commas):

- Manufacturer
- Model Number
- Serial Number (returns 0 if not available)
- Driver Revision (returns 0 if not available)

*IDN? returns the following response strings depending on model and options:
HEWLETT-PACKARD,E1413C,<*serial number*>,<*revision number*>
HEWLETT_PACKARD,E1313A_32CH,<*serial number*>,<*revision number*>
HEWLETT_PACKARD,E1313A_64CH,<*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.

*LMC?

Learn Macros Query. Returns a quoted string name for each currently defined macro. If more than one macro is defined, the strings are separated by commas (,). If no macro is defined, *LMC? returns a null string.

*OPC

Operation Complete. Causes an instrument to set bit 0 (Operation Complete Message) in the Standard Event Status Register when all pending operations 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 CALibration:SETup or CALibration:TARE commands have completed. Instead, use their query forms CALibration:SETup? or CALibration: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 CALibration:SETup or CALibration:TARE commands have completed. Instead, use their query forms CALibration:SETup? or CALibration:TARE?

*PMC

Purge Macros Command. Purges all currently defined macros.

*RMC <*name*>

Remove individual Macro Command. Removes the named macro command.

*RST

Reset Command. Resets the HP E1313/E1413 as follows:

- Sets all four scan lists to their default states:
  - Scan List 1= ROUTe:SEQuence:DEFine (@100:163),
    Scan List 2 through 4 are zero length (*undefined*)
  - SENSe:FUNCtion:VOLTage:DC AUTO,(@100:163) (all channels DCV, autorange)
- Sets the trigger system as follows:
  - TRIGger:SOURce HOLD
  - TRIGger:TIMer:PERiod 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
- Sets bit 13 in the Questionable Data Group (see Table 5-4 on page 259).

***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 Figure 4-6 for an illustration showing the contents of the Status Byte Group. 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 on page 277 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 Command. Triggers an instrument when the trigger source is set to bus (TRIGger:SOURce 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. Module must be screwed securely to mainframe.
3. 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 in this section. See the documentation that comes with the HP E1413 C-SCPI driver for MS-DOS®.

Comments

- This command will cause Error +3000, "illegal while initiated" if trigger is initiated.
- **Returned Value:**

Value	Meaning	Further Action
0	*TST? OK	None
-1	*TST? Error	Query the Error Queue (SYSTem:ERRor?) For Error +3052, see explanation below.

- IF Error +3052, "Self test failed. Test info in FIFO" is returned. A FIFO value of 1 through 99 is a failed test number. A value of 100 through 163 is a channel number for the failed test. A value of 200 through 204 is an A/D range number for the failed test where 200 = .0625, 201 = .25V, 202 = 1V, 203 = 4V, and 204 = 16V ranges. For example SENS:DATA:FIFO? returns the values 72 and 108. This indicates that test number 72 failed on channel 8.
- Test numbers 20, 30-37, 72, 74-76, and 80-93 may indicate a problem with a Signal Conditioning Plug-on.
- For tests 20, and 30-37, remove all SCPs and see if *TST? passes. If so, replace SCPs one at a time until you find the one causing the problem.
- For tests 72, 74-76, and 80-93, try to re-seat the SCP that the channel number(s) points to, or move the SCP and see if the failure(s) follow the SCP. If the problems move with the SCP, replace the SCP.
- These are the only tests where the user should troubleshoot a problem. Other tests which fail should be referred to qualified repair personnel.

Note Executing *TST? returns the module to its *RST state. *RST causes the FIFO data format to return to its default of ASC,7. If you want to read the FIFO for *TST? diagnostic information and you want that data in other than the ASCII,7 format, be certain to set the data FIFO format to the desired format (FORMat:DATA command) after completion of *TST? but before executing a SENSe:DATA:FIFO:… query command.

- The C-SCPI type for this returned value is **int16**.
- Following *TST?, the module is placed in the *RST state. This returns many of the module's programmed states to their defaults. See page 66 for a list of the module's default states.
- *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 and 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 multiplexer interface with EU CPU.
33,71	Checks opening of all relays on power down or input overvoltage.
34-37	Check FET multiplexer interface with A/D digital.

Analog Tests:

Test#	Description
40-42	Checks internal voltage reference.
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.
80	Checks mid-scale strain DAC output. Only reports first channel of SCP.
81	Checks range of strain DAC. Only reports first channel of SCP.
82	Checks noise of strain DAC. Only reports first channel of SCP.
83	Checks bridge completion leg resistance each channel.
84	Checks combined leg resistance each channel.
86	Checks current source SCP's OFF current.
87	Checks current source SCP's current dac mid-scale.
88	Checks current source SCP's current dac range on HI and LO ranges.
89	Checks strain SCP's Wagner Voltage control.
91	Checks autobalance DAC range with input shorted.
92	Sample and Hold channel holds value even when input value changed.
93	Sample and Hold channel held value test for droop rate.

*WAI

Wait-to-continue Command. 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 CALibration:SETup or CALibration:TARE commands have completed. Instead, use their query forms CALibration:SETup? or CALibration:TARE?. CALibration:SETup? and CALibration:TARE? return a value only after the CALibration:SETup or CALibration:TARE operations are complete.

Command Quick Reference

The following tables summarize SCPI and IEEE-488.2 Common (*) Commands for the HP E1313/E1413 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:SOURce 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 INITiated.
:CURRent?	Returns composite limit test status for last completed scan list.
:FLIMits	
[:CHANnels]	
[:CUMulative]?	Returns all channel's limit test status since module was INITiated.
:CURRent?	Returns all channel's limit test status for last completed scan list.
:POINts	
[:CUMulative]?	Returns count of channels exceeding limit tests since module INITiated.
:CURRent?	Returns count of channels exceeding limit tests for last completed scan.
:LIMit	
[:STATe] ON | OFF,(@<*ch_list*>)	Enables/disables all limit testing.
[:STATe]? (@<*channel*>)	Returns state of limit testing.
:FAIL	
[:CUMulative]? (@<*channel*>)	Returns limit test status for *channel* since module was INITiated.
:CURRent? (@<*channel*>)	Returns limit test status for *channel* for last completed scan list.
:LOWer	
[:STATe] ON | OFF,(@<*ch_list*>)	Enables/disables 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/disables upper limit testing.
[:STATe]? (@<*channel*>)	Returns state of upper limit testing.
:DATA <*upper_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 CALibration:SETup operation (returns error codes or 0 for OK).
:STORe ADC | TARE	Store cal constants to Flash RAM for either A/D calibration or those generated by the CALibration:TARE command.

SCPI Command Quick Reference	
Command	**Description**
:TARE (@<*ch_list*>)	Calibrate out system field wiring offsets.
:RESet	Resets cal constants from CALibration:TARE back to zero for all channels.
:TARE?	Returns state of CALibration:TARE operation (returns error codes or 0 for OK).
:VALue	
:RESistance <*ref_ohms*>	Send to instrument the value of just measured reference resistor.
:VOLTage <*ref_volts*>	Send to instrument the value of just measured voltage reference.
:ZERO?	Correct A/D for short term offset drift (returns error codes or 0 for OK).
DIAGnostic	
:CALibration	
:TARE	
[:OTDetect]	
[:MODE]	Sets mode to control OTD current during tare calibration.
[:MODE]?	Returns current setting of OTD control during tare calibration.
:CHECksum?	Perform checksum on Flash RAM and return a '1' for OK, a '0' for corrupted or deleted memory contents.
:COMMand	
:SCPWRITE <*reg_addr*>,<*reg_data*>	Writes values to SCP registers.
:CUSTom	
:LINear <*table_ad_range*>,<*table_block*>,(@<*ch_list*>)	Loads linear custom EU table.
:PIECewise <*table_ad_range*>,<*table_block*>,(@<*ch_list*>)	Loads piecewise custom EU table.
:REFerence:TEMPerature	Puts the contents of the Reference Temperature Register into the FIFO.
:FLOor[:CONFigure] <*range*>,(@<*ch_list*>)	Sets lowest range that autorange can select.
:FLOor:DUMP	Sends the autorange floor setting for all 64 channels to the FIFO.
:INTerrupt[:LINe] <*intr_line*>	Sets the VXIbus interrupt line the module will use.
:INTerrupt[:LINe]?	Returns the VXIbus interrupt line the module is using.
:OTDetect[:STATe] ON | OFF, (@<*ch_list*>)	Controls "Open Transducer Detect" on SCPs contained in <*ch_list*>.
:OTDetect[:STATe]? (@<*channel*>)	Returns current state of OTD on SCP containing <*channel*>.
:QUERy	
:SCPREAD <*reg_addr*>	Returns value from an SCP register.
:VERSion?	Returns manufacturer, model, serial#, flash revision #, and date e.g. HEWLETT-PACKARD,E1413C,US34000478,A.04.00, Wed Jul 08 11:06:22 MDT 1994
FETCh?	Return readings stored in VME Memory (format set by FORMat:DATA cmd).
FORMat	
[:DATA] <*format*>[, <*size*>]	Set format for response data from [SENSe:]DATA?.
ASCii[, 7]	Seven bit ASCII format (not as fast as 32-bit because of conversion).
PACKed[, 64]	Same as REAL, 64 except NaN, +INF, and -INF format compatible with HP BASIC.
REAL[, 32]	IEEE 32-bit floating point (requires no conversion so is fastest).
REAL, 64	IEEE 64-bit floating point (not as fast as 32-bit because of conversion).
[:DATA]?	Returns format: REAL, +32 | REAL, +64 | PACK, +64 | ASC, +7.
INITiate	
: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 Wait-for-Trigger state (ready to make one scan).
INPut	
:FILTer	Control filter Signal Conditioning Plug-ons.
[:LPASs]	
:FREQuency <*cutoff_freq*<*D*>,(@<*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**
:LOW <*wvolt_type*>,(@<*ch_list*>)	Controls the connection of input LO on a Strain Bridge (Opt. 21 SCP).
:LOW? (@<*channel*>)	Returns the LO connection for the Strain Bridge at *channel*.
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 INITiate:IMMediate.
: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.
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 LIST*n* or all Scan Lists.
:POINts? LIST1 | LIST2 | LIST3 | LIST4 | LISTL	Returns the number of points (channels) in Scan LIST*n*.
SAMPle	
:TIMer LIST1 | LIST2 | LIST3 | LIST4 | LISTL | ALL,<*interval*>	Sets the time interval in seconds between channel measurements for Scan LIST*n*, or all four Scan Lists.
:TIMer? LIST1 | LIST2 | LIST3 | LIST4 | LISTL	Returns the timer interval in seconds for Scan LIST*n*.
[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 12 KHz low-pass filter.
[:STATe]?	Returns the state of the A/D's low-pass filter.

SCPI Command Quick Reference	
Command	**Description**
FUNCtion	Equate a function and range with groups of channels.
:CUSTom [<*range*>,](@<*ch_list*>)	Links channels to custom EU conversion table loaded by DIAGnostic:CUSTom:LINear or DIAGnostic:CUSTom:PIECewise commands.
:REFerence [<*range*>,](@<*ch_list*>)	Links channels to custom reference temperature EU conversion table loaded by DIAGnostic:CUSTom:PIECewise commands.
:TC <*type*>,[<*range*>,](@<*ch_list*>)	Links channels to custom temperature EU conversion table loaded by DIAGnostic:CUSTom:PIECewise, 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	two-wire RTDs
TCouple, CUST | E | EEXT | J | K | N | S | T	thermocouples
THERmistor, 2250 | 5000 | 10000	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	RTDs
THERmistor,5000	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, 11=Limit Exceeded
:CONDition?	Returns state of Operation Status signals.
:ENABle <*enable_mask*>	Bits set to 1 enable status events to be summarized into Status Byte.
:ENABle?	Returns the decimal weighted sum of bits set in the Enable Register.
[:EVENt]?	Returns weighted sum of bits that represent Operation status events.
:NTRansition <*transition_mask*>	Sets mask bits in the Negative Transition Filter (NTF) Register.
:NTRansition?	Returns value of bits set in the NTF Register.
:PTRansition <*transition_mask*>	Sets mask bits in the Positive Transition Filter (PTF) Register.
:PTRansition?	Returns value of bits set in the PTF Register.
: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.

SCPI Command Quick Reference	
Command	**Description**
[:EVENt]?	Returns weighted sum of bits that represent Questionable Data events.
:NTRansition <*transition_mask*>	Sets mask bits in the Negative Transition Filter (NTF) Register.
:NTRansition?	Returns value of bits set in the NTF Register.
:PTRansition <*transition_mask*>	Sets mask bits in the Positive Transition Filter (PTF) Register.
:PTRansition?	Returns value of bits set in the PTF Register.
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 TRIGger:SOURce 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 TRIGger:SOURce is TIMer
:MODE ASYNchronous | SYNChronous	ASYN; Trigger Timer runs only while module is INITiated, SYNC; Trigger Timer runs continuously.
:MODE?	Returns setting of Trigger Timer Mode.
[:PERiod] <*trig_interval*>	Sets the interval between scan triggers when TRIGger:SOURce 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,E1413C,<*serial#*>,<*driver rev#*>.
	*RST	Reset	Resets all scan lists to zero length and stops scan triggering. Status registers and output queue are unchanged.
	*TST?	Self Test	Performs self test. Returns 0 to indicate test passed.
Status Reporting	*CLS	Clear Status	Clears all status event registers and so their status summary bits (except the MAV bit).
	*ESE <*mask*>	Event Status Enable	Set Standard Event Status Enable register bits mask.
	*ESE?	Event Status Enable query	Return current setting of Standard Event Status Enable Register.
	*ESR?	Event Status Register query	Return Standard Event Status Register contents.
	*SRE <*mask*>	Service Request Enable	Set Service Request Enable register bit mask.
	*SRE?	Service Request Enable query	Return current setting of the Service Request Enable register.
	*STB?	Read Status Byte query	Return current Status Byte value.
Macros	*DMC <*name*>,<*cmd_data*>	Define Macro Command	Assigns one, or a sequence of commands to a macro.
	*EMC 1 | 0	Enable Macro Command	Enable/Disable defined macro commands.
	*EMC?	Enable Macros query	Returns 1 for macros enabled, 0 for disabled.
	*GMC? <*name*>	Get Macro query	Returns command sequence for named macro.
	*LMC?	Learn Macro query	Returns comma-separated list of defined macro names.
	*PMC	Purge Macro Commands	Purges all macro commands
	*RMC <*name*>	Remove Individual Macro	Removes named macro command.
Synchronization	*OPC	Operation Complete	Standard Event register's Operation Complete bit will be 1 when all pending device operations have been finished.
	*OPC?	Operation Complete query	Places an ASCII 1 in the output queue when all pending operations have finished.
	*TRG	Trigger	Triggers module when TRIGger:SOURce is HOLD.
	*WAI	Wait to Complete	

Notes

Chapter 6
Signal Conditioning Plug-on Manuals

Place your SCP manuals in this section

Notes

Appendix A
Specifications

Except where noted, all references to the HP E1413 apply to the HP E1313. 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.

Power Requirements

	+5V		+12V		-12V		+24V		-24V		-5.2V	
IPm=Peak Module Current	IP_m	ID_m	IP_m	IDm	IP_m	ID_m	IP_m	ID_m	IP_m	ID_m	IP_m	ID_m
IDm=Dynamic Module Current	0.20	0.	0.06	0.01	0.01	0.01	0.5	0.1	0.5	0.1	0.4	0.1

Power Available for SCPs
(See the VXI Catalog or SCP User's Manual for SCP current)

HP E1313A: .55A ±24V, 3.5A 5V

HP E1413C: 1A ±24V, 3.5A 5V

Cooling Requirements

Average Watts/Slot	ΔPressure (mmH_2O)	Air Flow (liters/s)
34	0.08	0.08

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

[SENSe:]FILT OFF: 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

[SENSe:]FILT ON: 6.4K rdgs/sec divided by the number of channels in the scan - for example,
6.4K rdgs/sec ÷ 64 channels = 100 rdgs/sec/ch
6.4K rdgs/sec ÷ 16 channels = 400 rdgs/sec/ch

Trigger Timer and Sample Timer Accuracy

100ppm (.01%) from -10 °C to +70 °C

External Trigger Input

TTL compatible input. Negative true edge triggered except first trigger will occur if external trigger input is held low when module is INITiated. Minimum pulse width 100nS. Since each trigger starts a complete scan of 2 or more channel readings, maximum trigger rate depends on module configuration.

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.). **For HP E1313, multiply Noise Spec. by 1.4.**

NOTE: When autorange in ON:
for readings < 3.8V, add ± 0.02% to linearity specifications
for readings ≥ 3.8V, add ± 0.05% to linearity specifications

A/D range ±V F.Scale	Linearity % of reading	Offset Error	Noise 3 sigma	Noise* 3 sigma
.0625	0.01%	5.3 µV	18 µV	8 µV
.25	0.01%	10.3 µV	45 µV	24 µV
1	0.01%	31 µV	110 µV	90 µV
4	0.01%	122 µV	450 µV	366 µV
16	0.01%	488 µV	1.8 mV	1.5 mV

* [SENSe:]FILTer[:LPASs][:STATe] ON (min sample time - 145µsec)

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

Temperature Accuracy

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

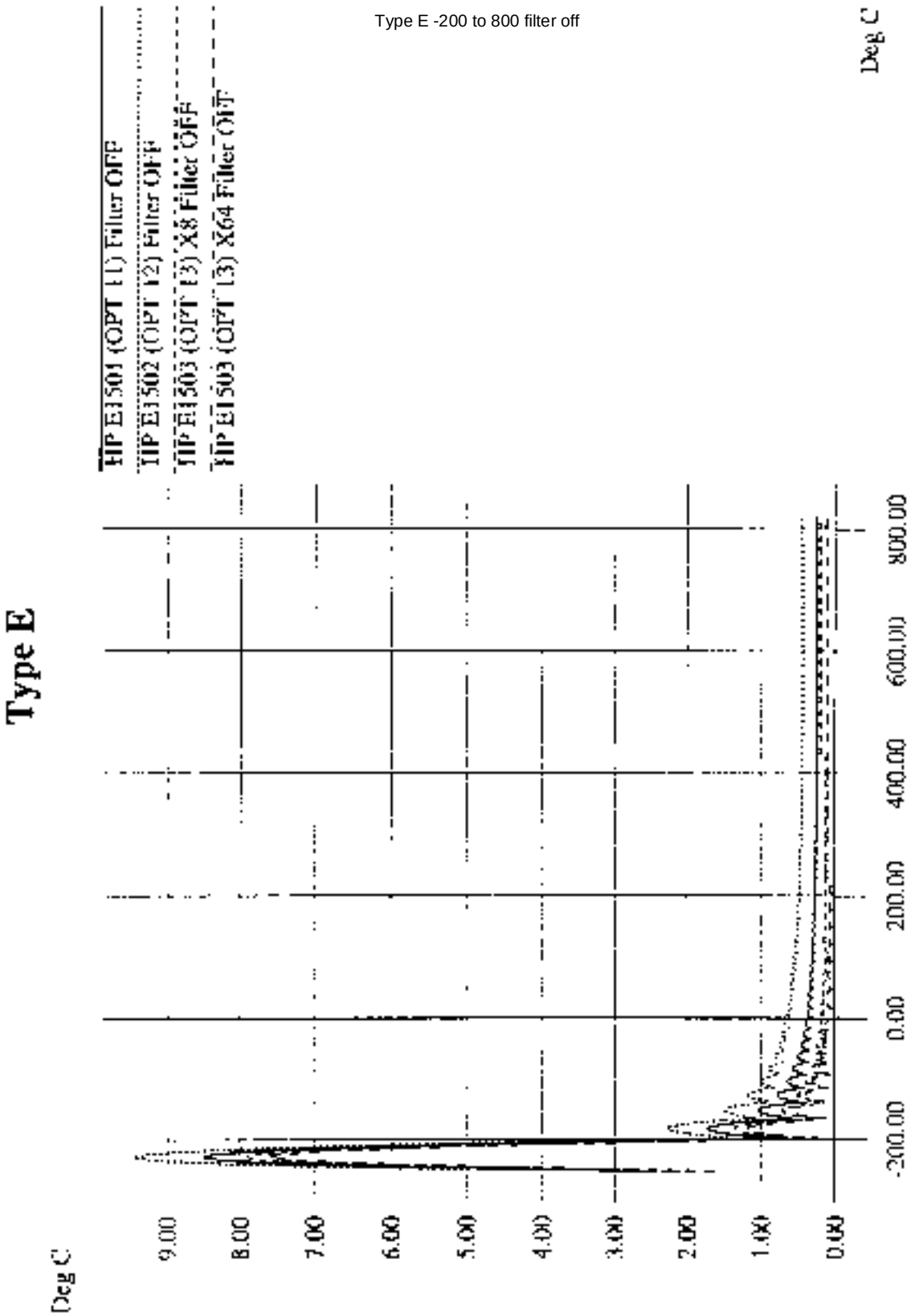
The thermocouple graphs on the following pages include only the errors due to measuring the voltage output of the thermocouple, as well as the algorithm errors due to converting the thermocouple voltage to temperature. To this error must be added the error due to measuring the reference junction temperature with an RTD or a 5K thermistor. See the graphs for the RTD or the 5K thermistor to determine this additional error. Also, the errors due to gradients across the isothermal reference must be added. If an external isothermal reference panel is used, consult the manufacturer's specifications. If HP termination blocks are used as the isothermal reference, see the notes below.

NOTE

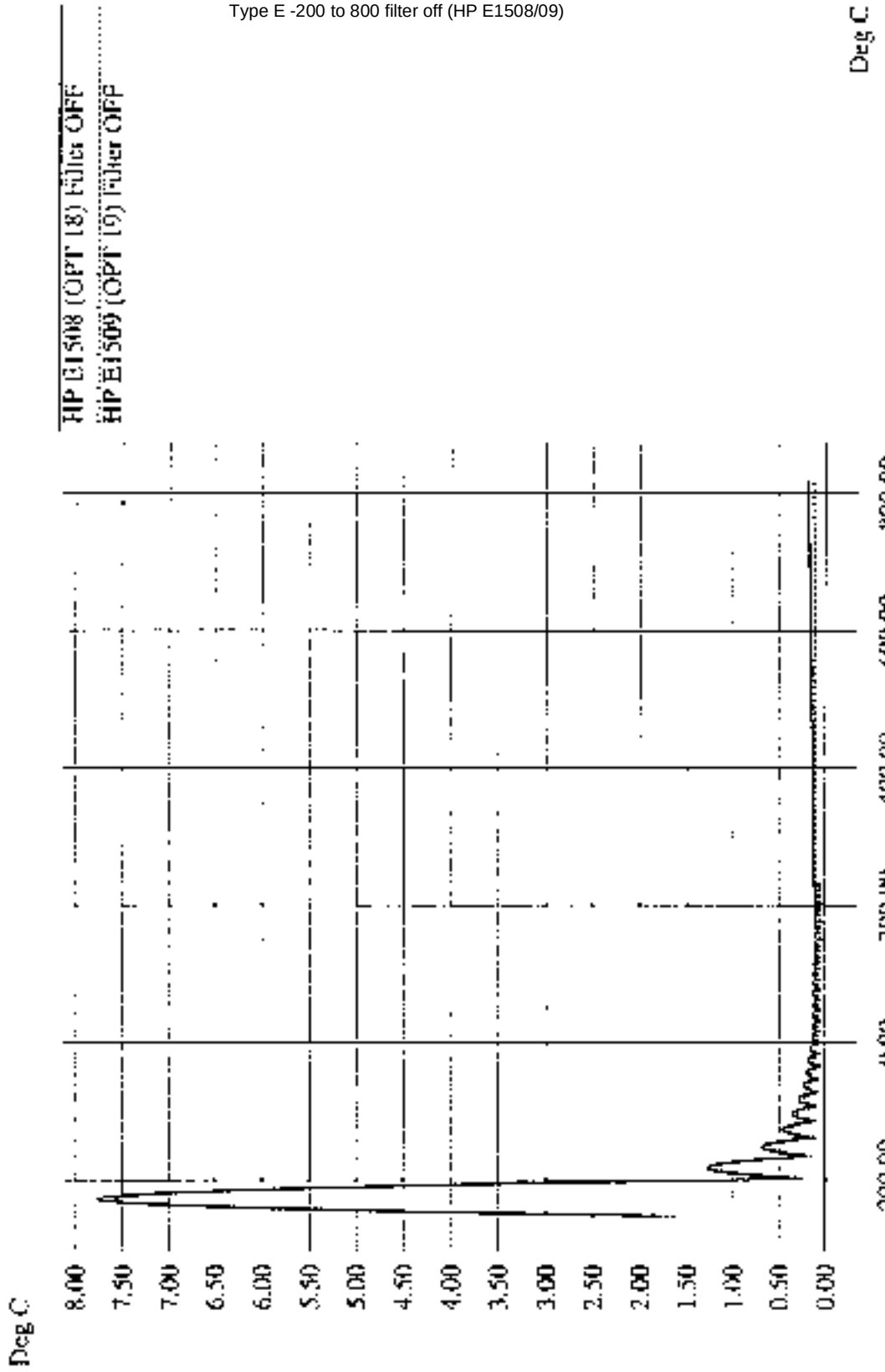
1) When using the Terminal Module as the isothermal reference, add ±0.6° C to the thermocouple accuracy specs to account for temperature gradients across the Terminal Module. The ambient temperature of the air surrounding the Terminal Module must be within ±2° C of the temperature of the inlet cooling air to the VXI mainframe.

2) When using the E1586 Rack-Mount Terminal Panel as the isothermal reference, add ±0.2° C to the thermocouple accuracy specs to account for temperature gradients across the E1586. The E1586A should be mounted in the bottom part of the rack, below and away from other heat sources for best performance.

The temperature graphs are found on the following pages:

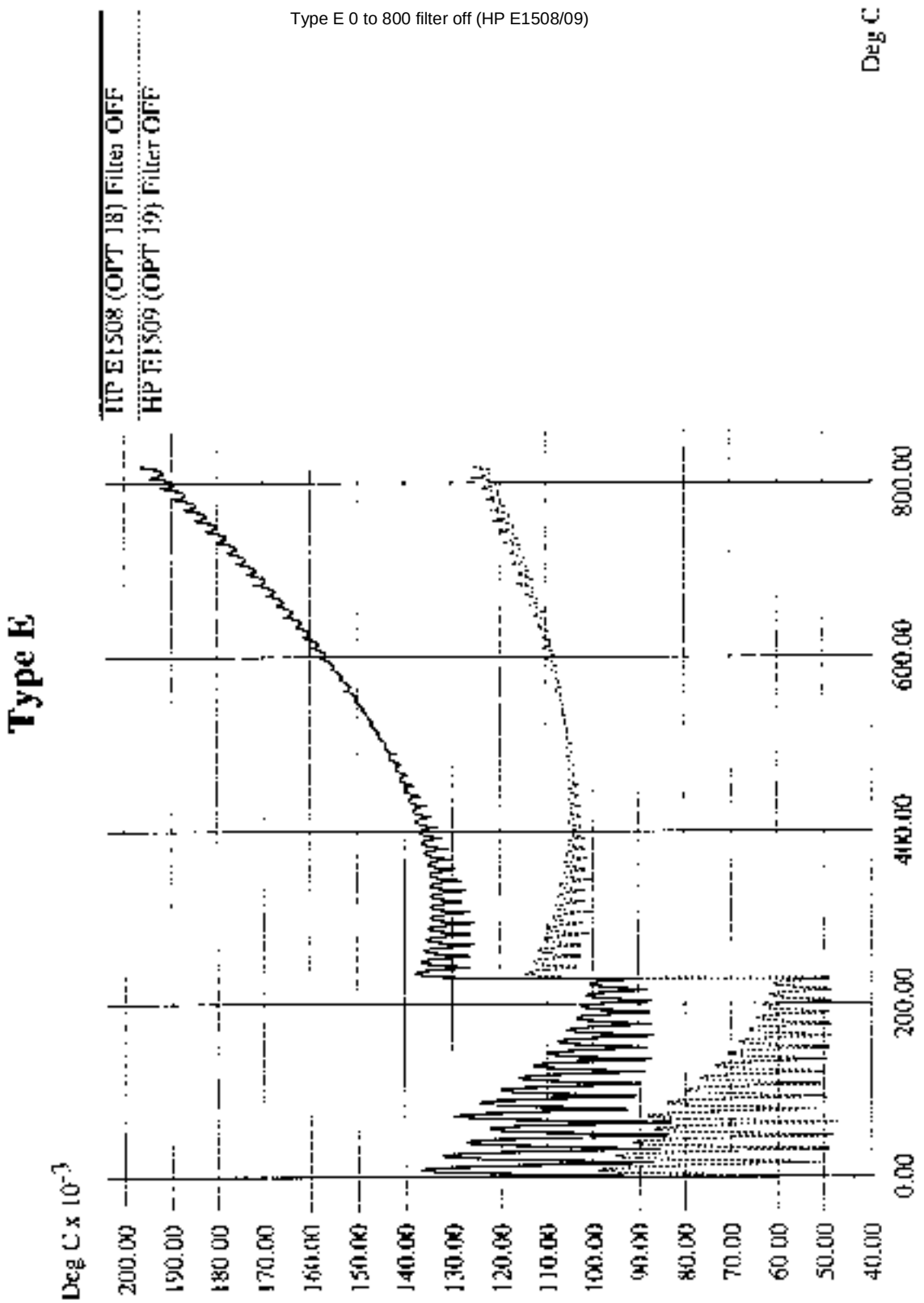

Type E -200 to 800 filter off
Type E
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C
Deg C
9.00
8.00
7.00
6.00
5.00
4.00
3.00
2.00
1.00
0.00
-200.00
0.00
200.00
400.00
600.00
800.00

Type E -200 to 800 filter off (HP E1508/09)

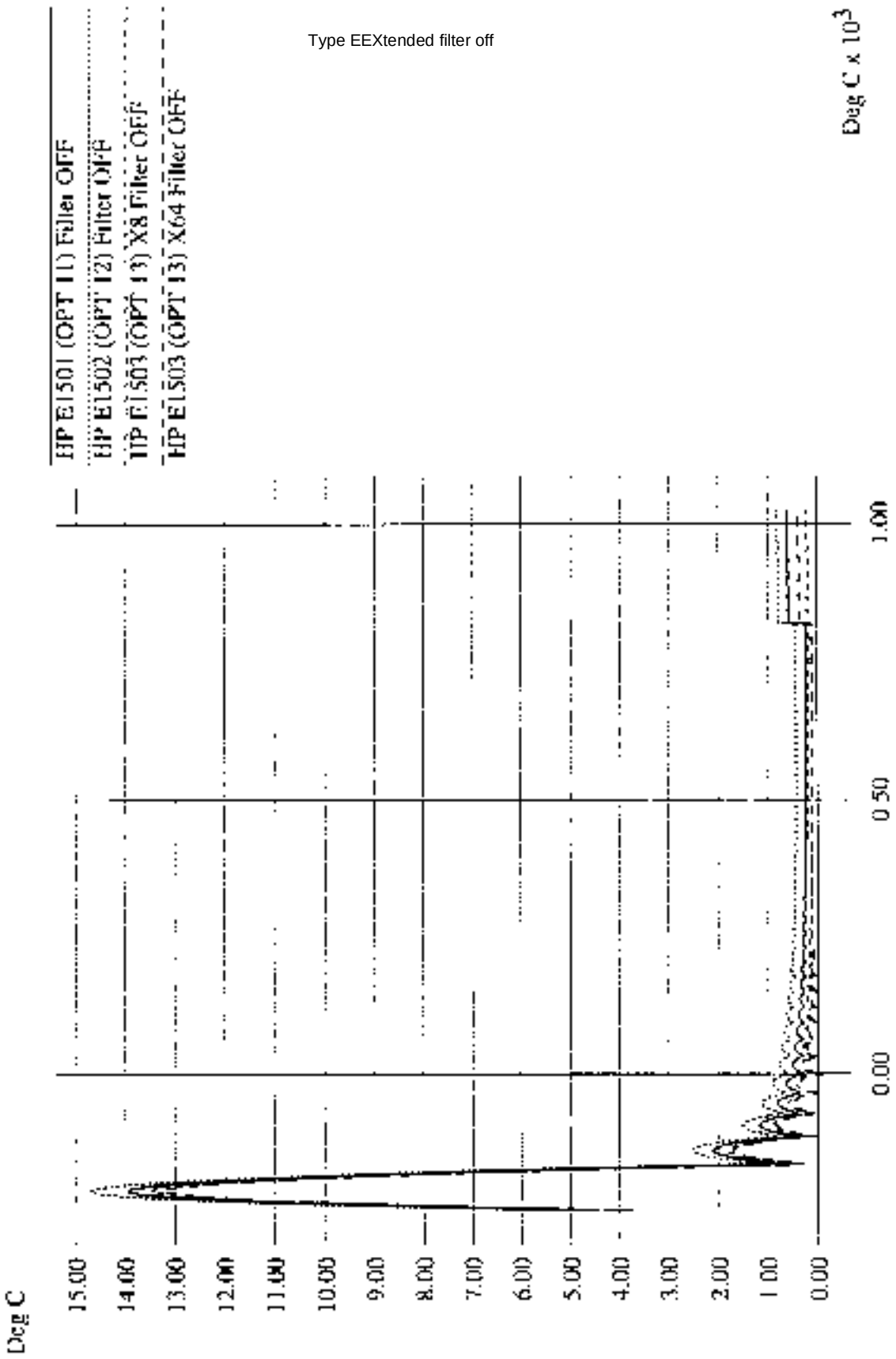


Type E
Type E 0 to 800 filter off
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C x 10⁻³
650.00
600.00
550.00
500.00
450.00
400.00
350.00
300.00
250.00
200.00
150.00
100.00
50.00
0.00
200.00
400.00
600.00
800.00
Deg C

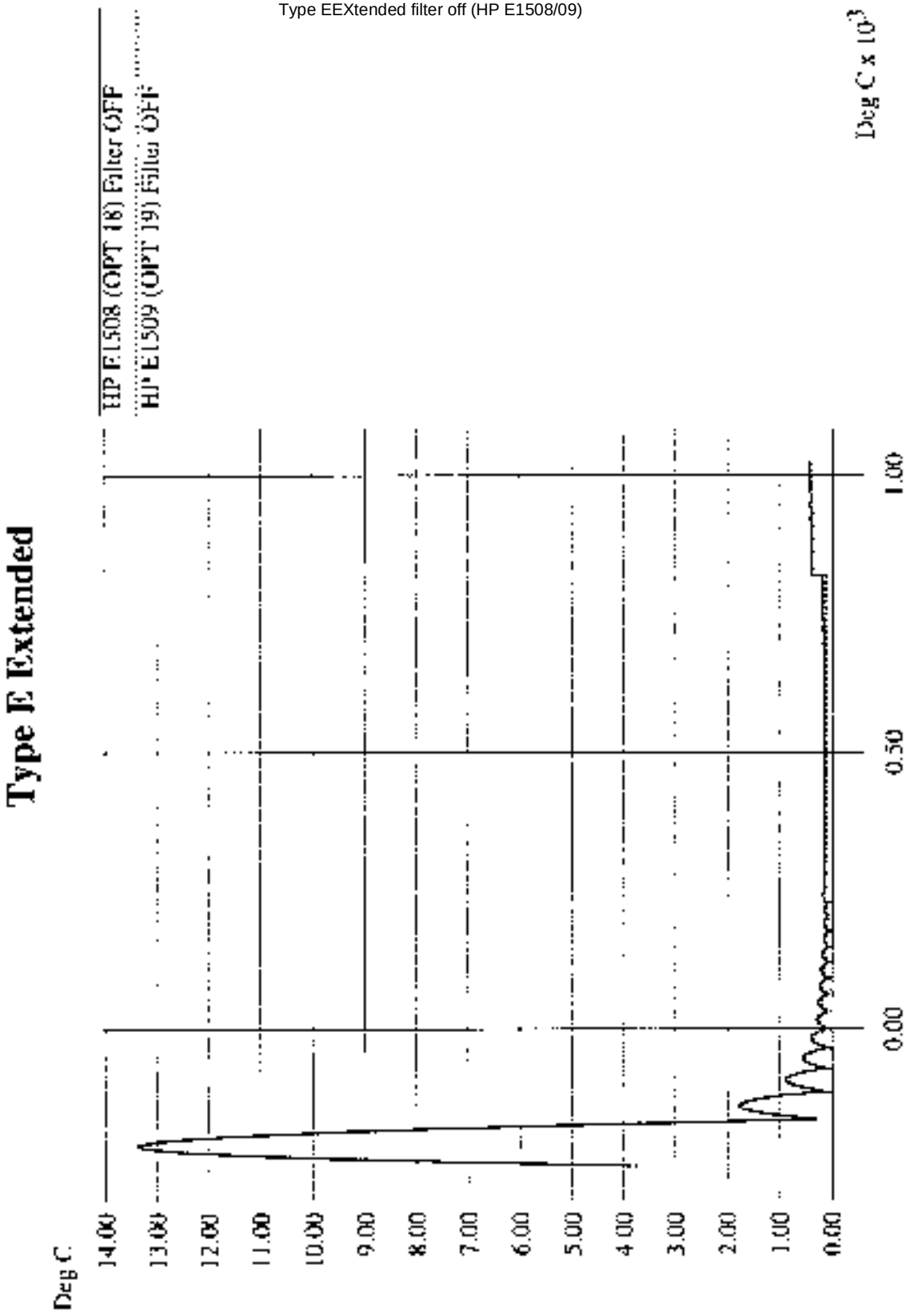
Type E 0 to 800 filter off (HP E1508/09)



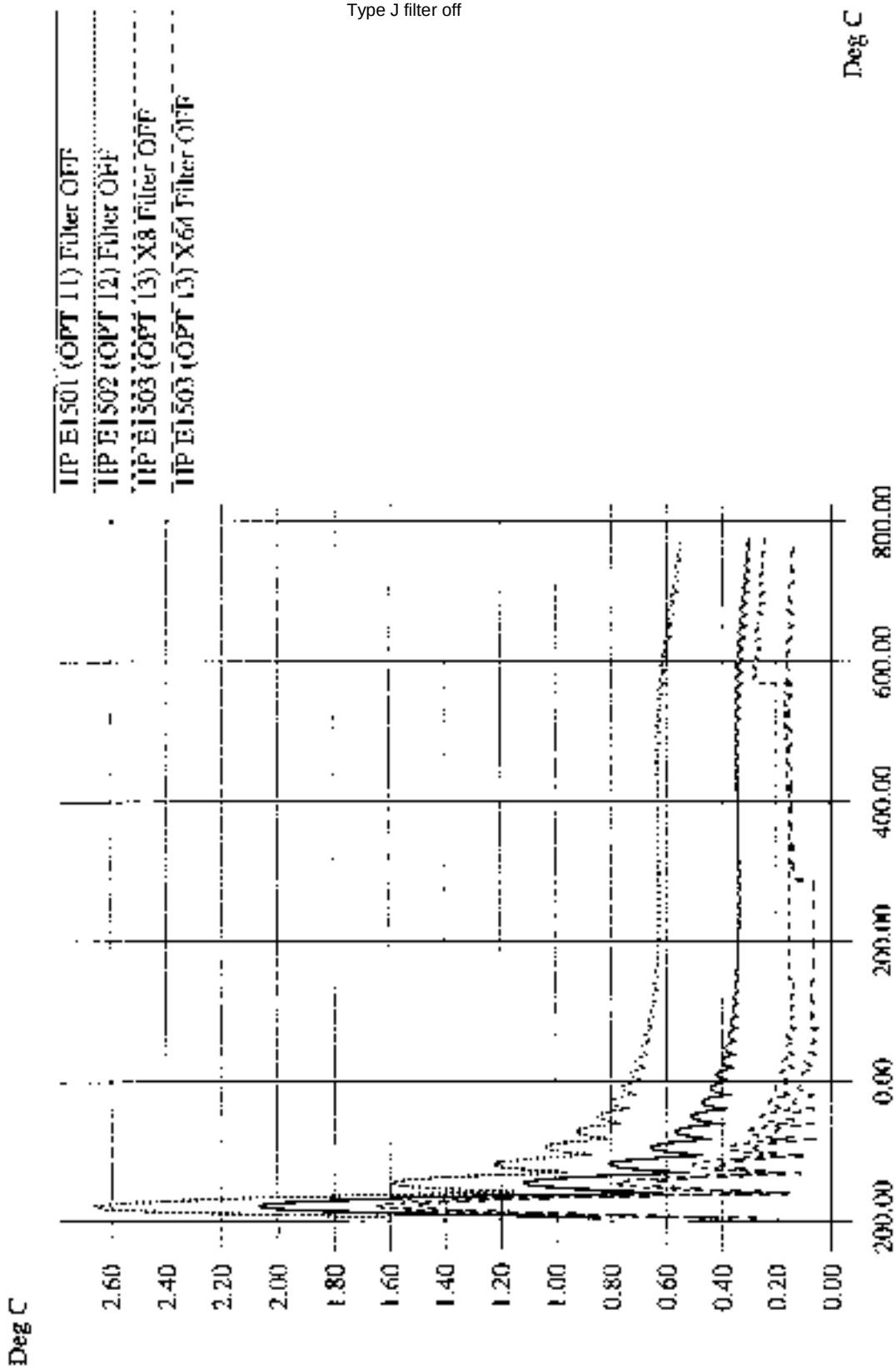
Type EEXtended filter off



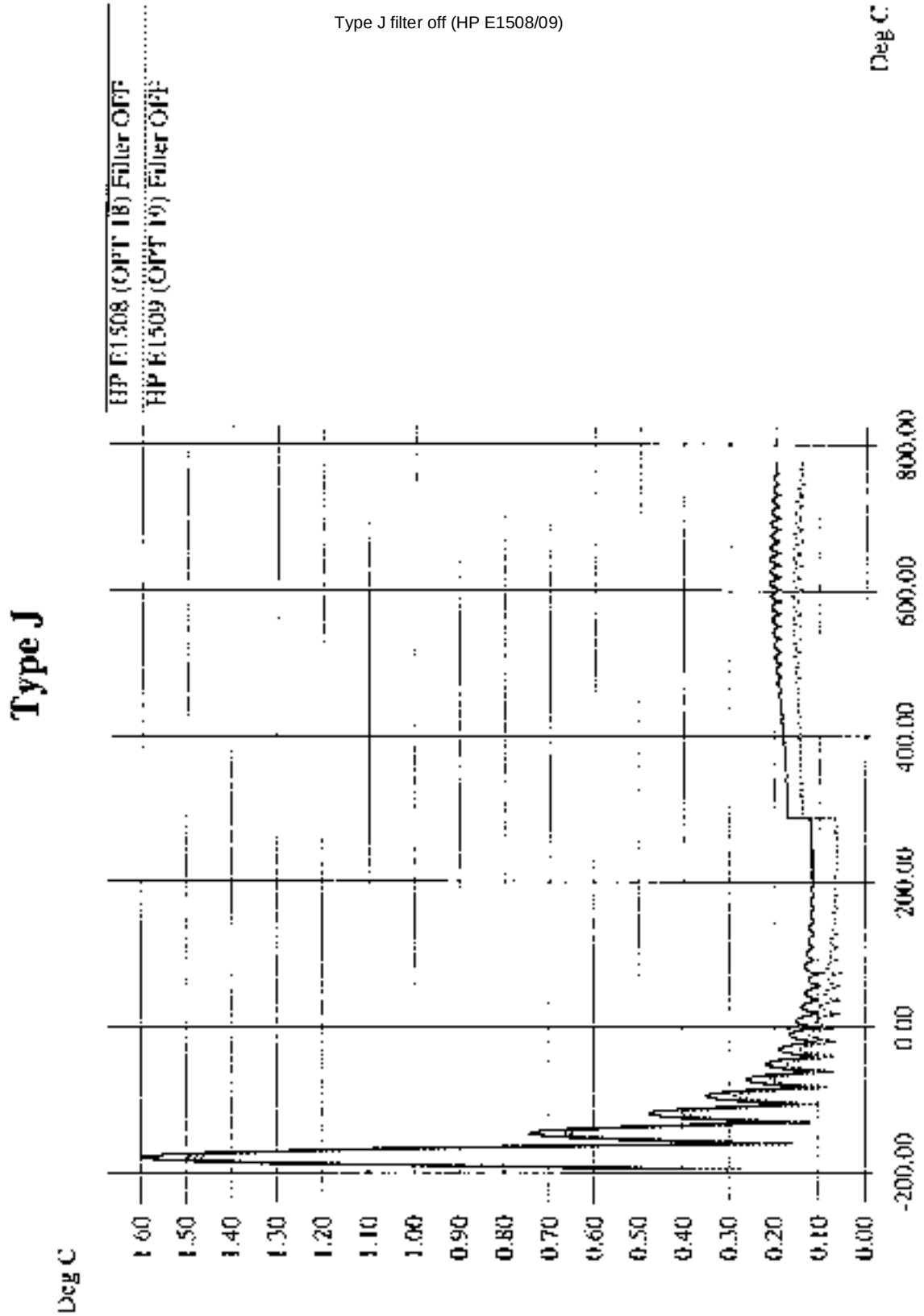
Type EExtended filter off (HP E1508/09)



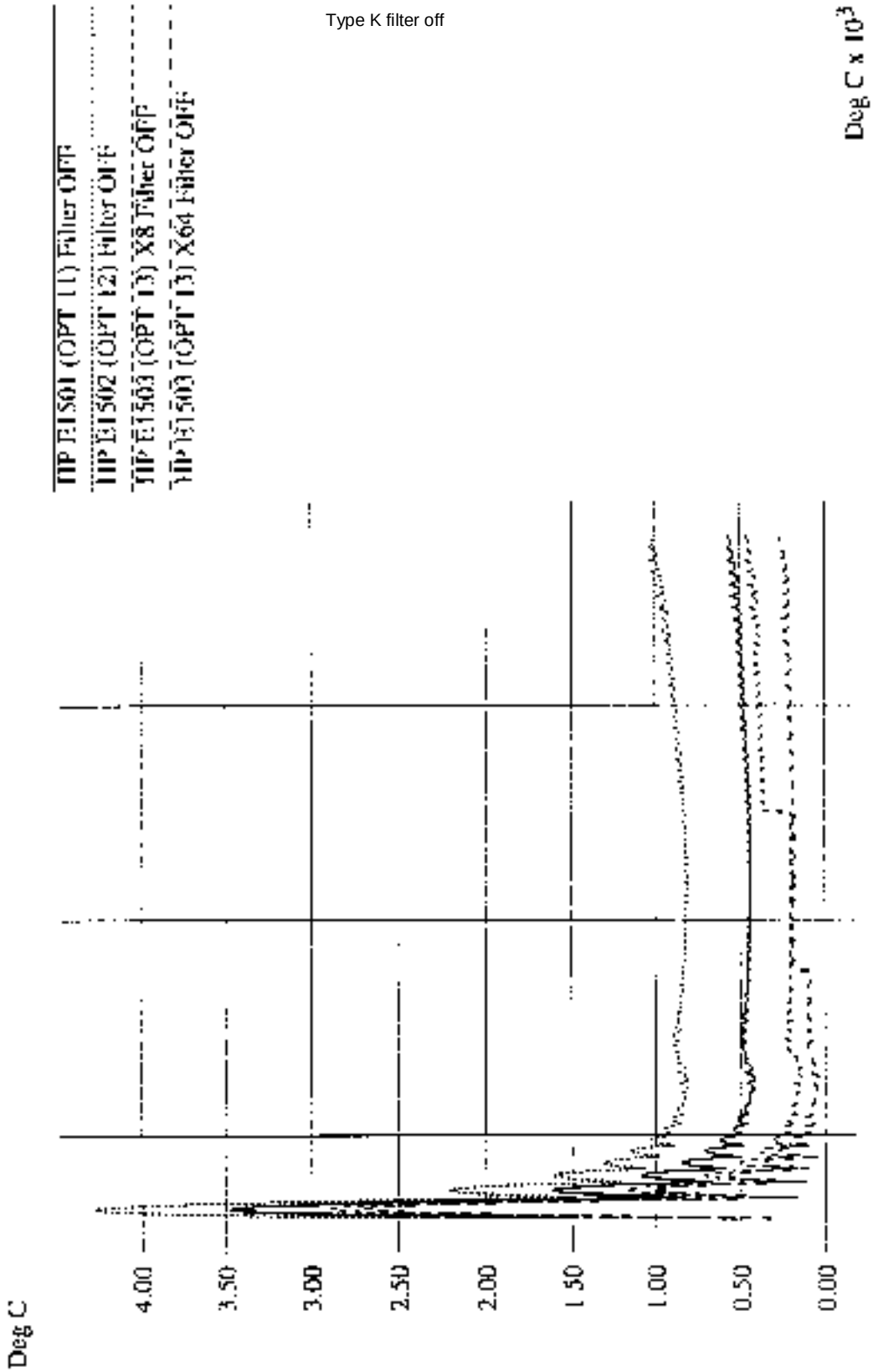
Type J filter off



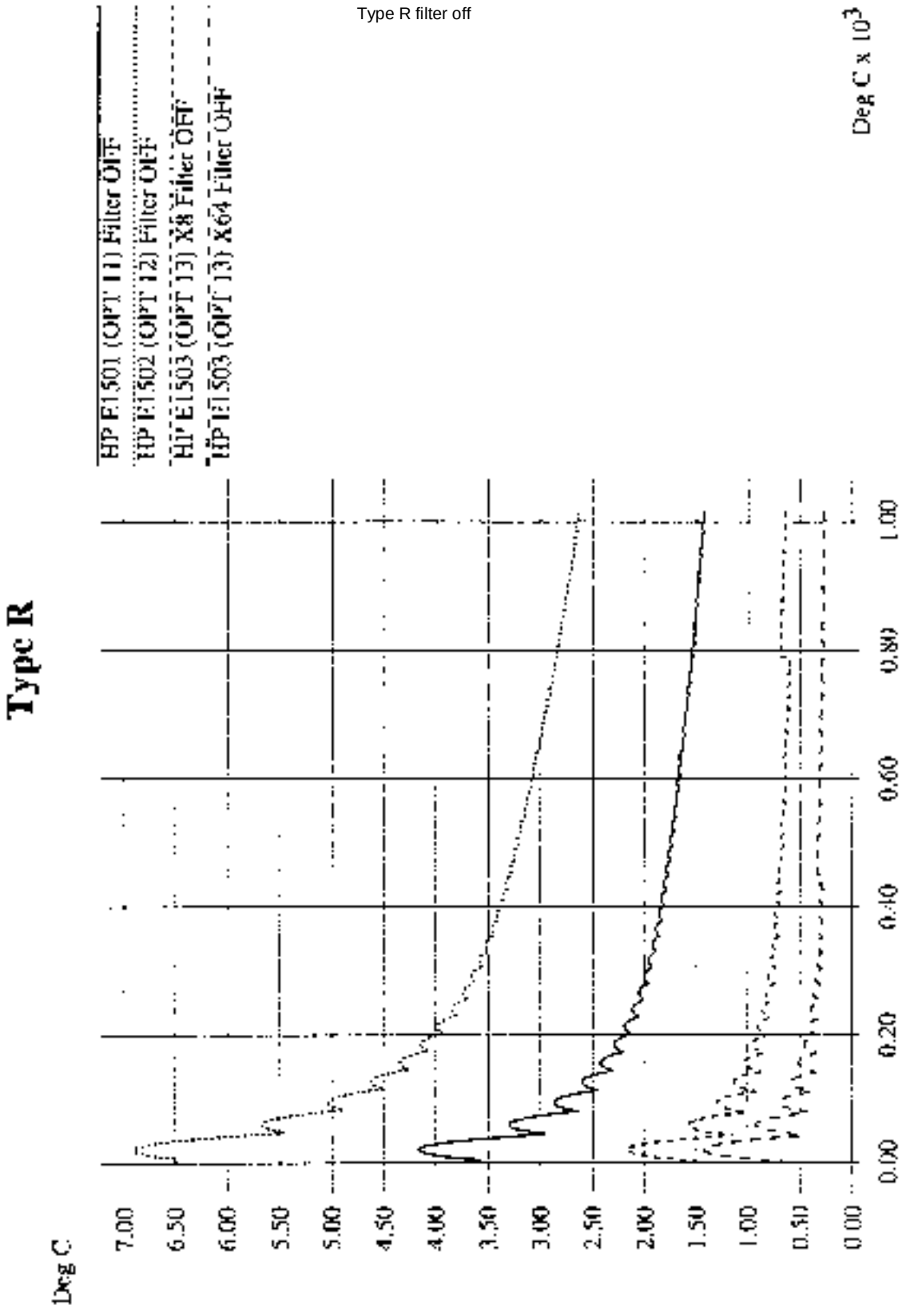
Type J filter off (HP E1508/09)



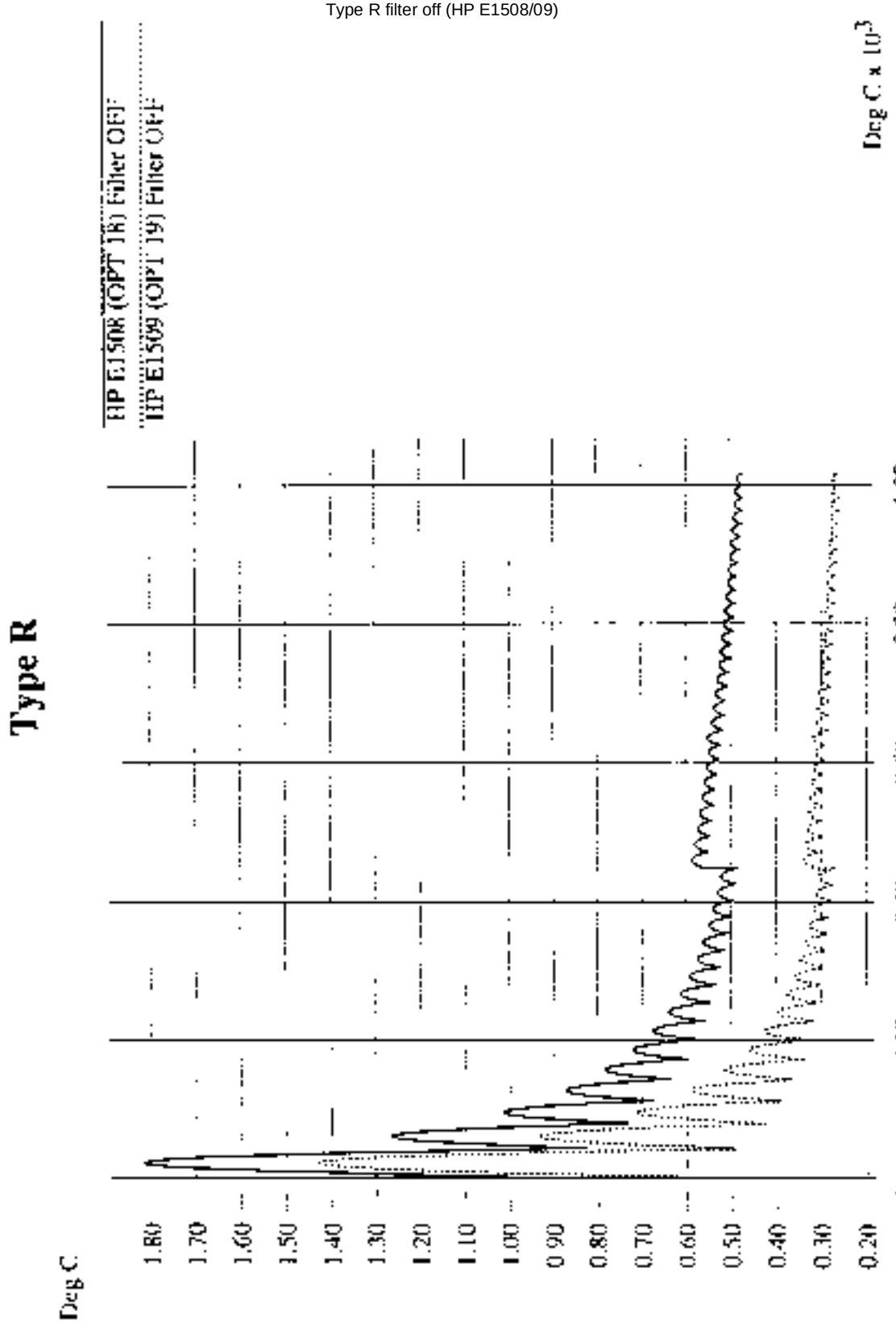
Type K filter off



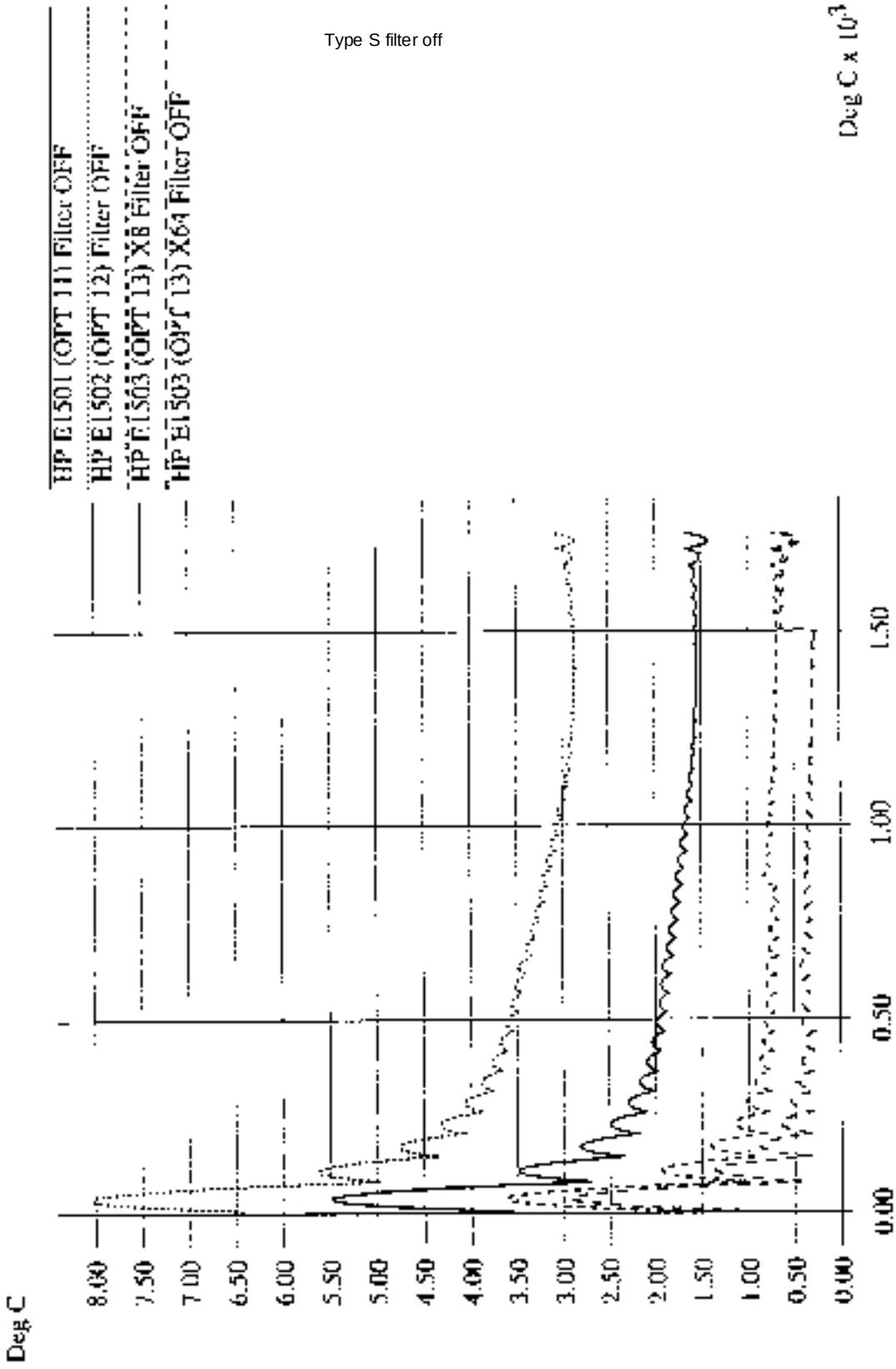
Type R filter off



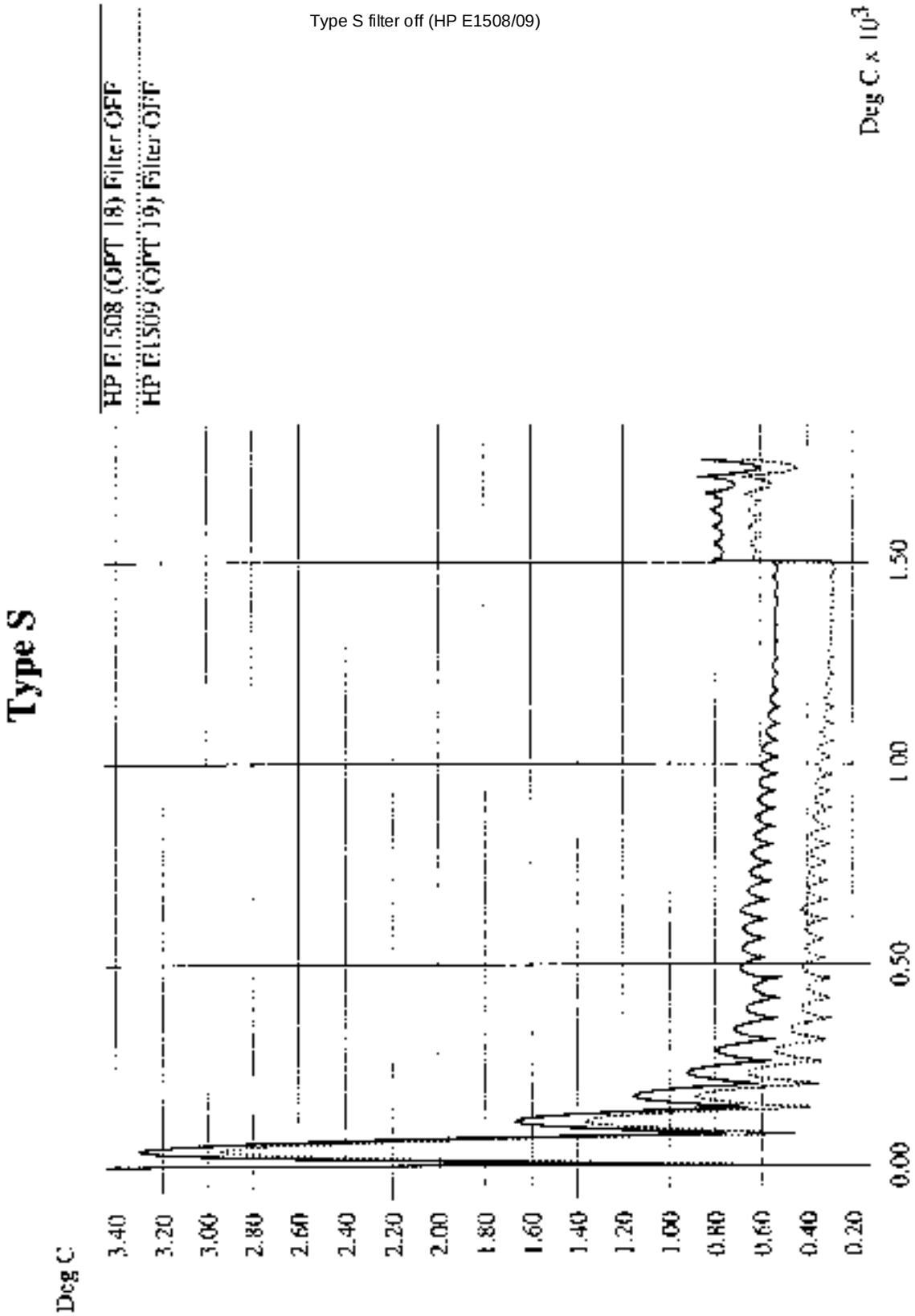
Type R filter off (HP E1508/09)

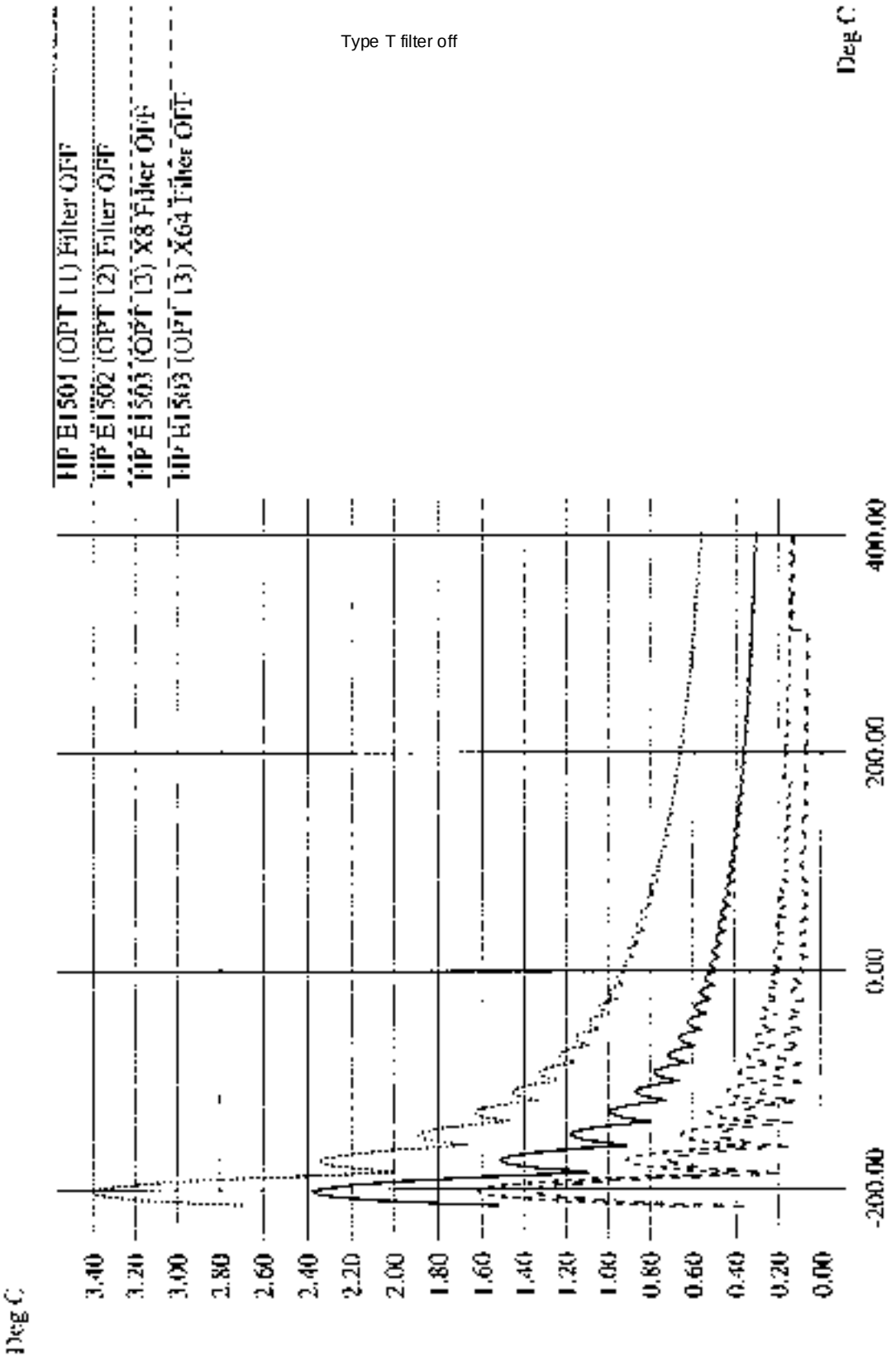


Type S filter off

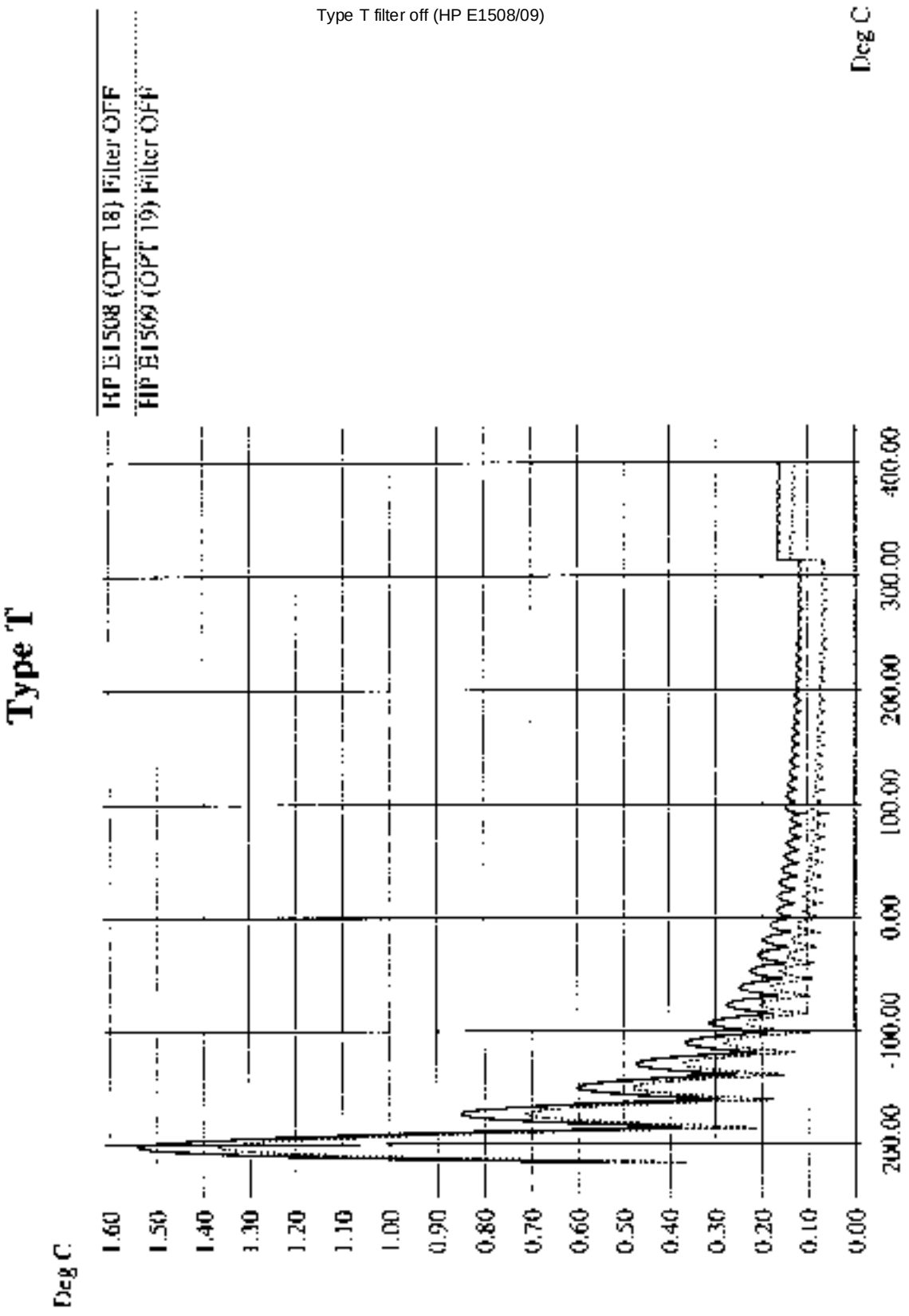


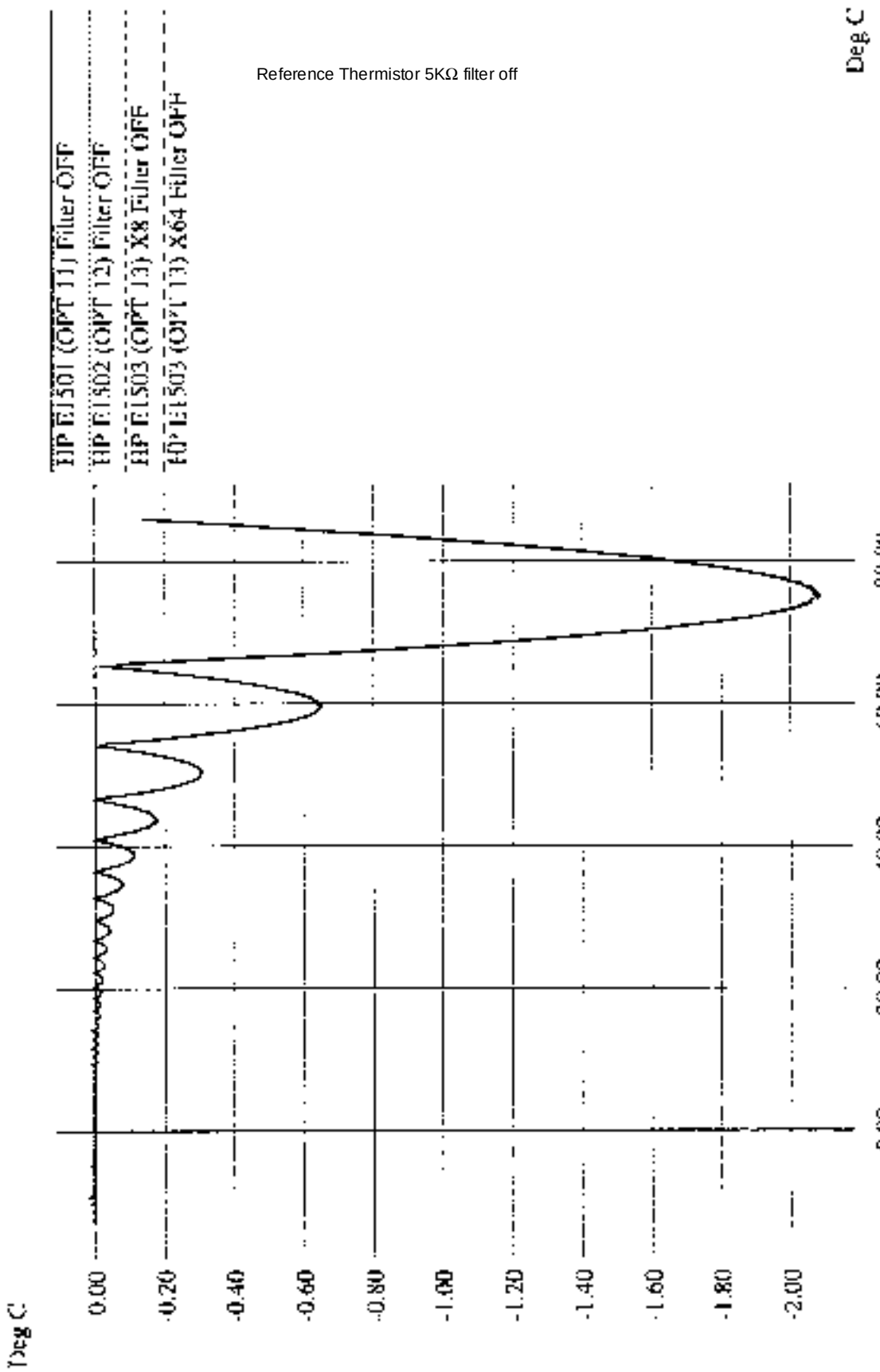
Type S filter off (HP E1508/09)



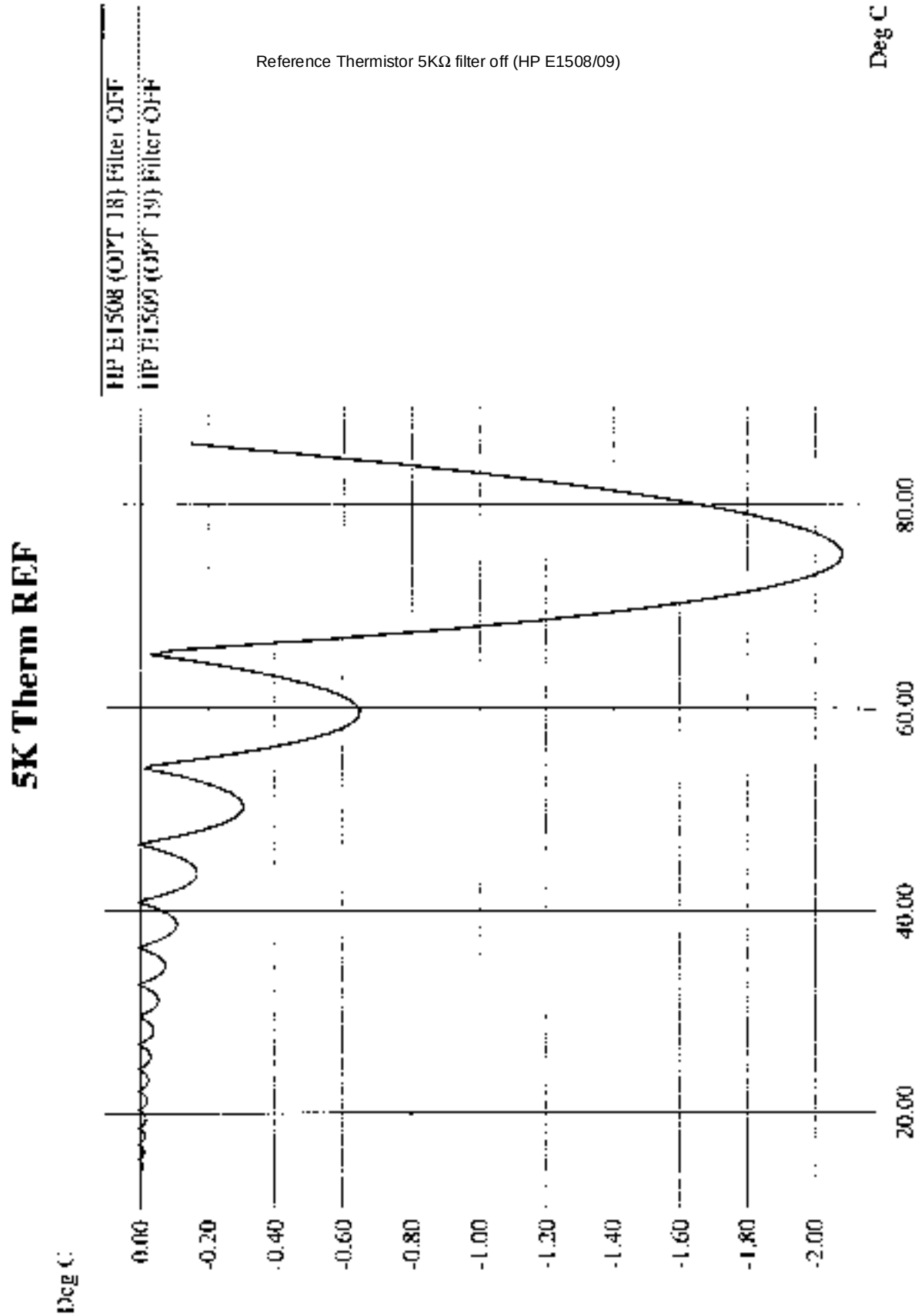

Type T
Type T filter off
HP E1501 (OPT 1) Filter OFF
HP E1502 (OPT 2) Filter OFF
HP E1503 (OPT 3) X8 Filter OFF
HP E1503 (OPT 3) X64 Filter OFF
Deg C
Deg C
3.40
3.20
3.00
2.80
2.60
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
0.60
0.40
0.20
0.00
-200.00
0.00
200.00
400.00

Type T filter off (HP E1508/09)

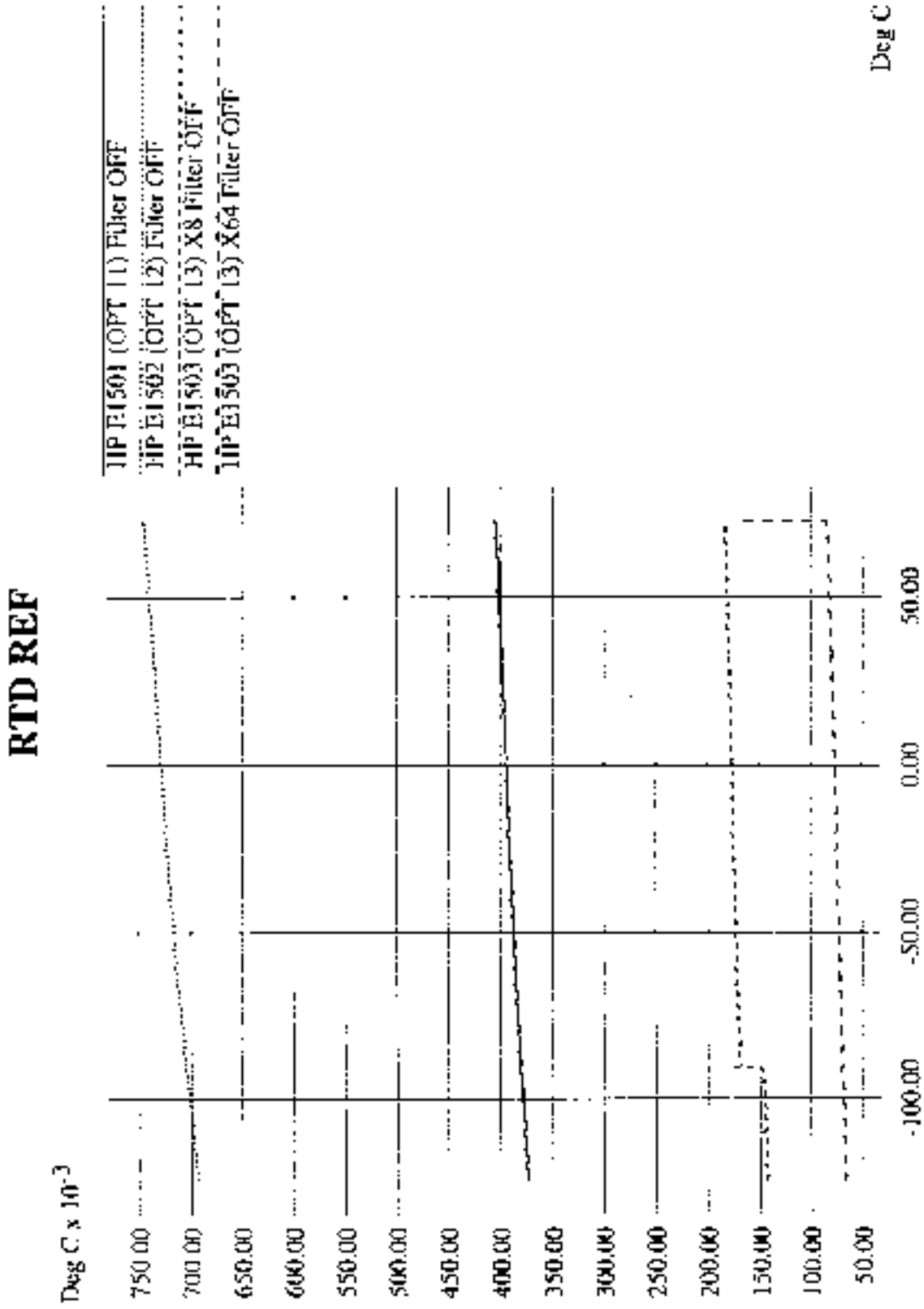



5K Therm REF
Reference Thermistor 5KΩ filter off
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C
Deg C
0.00
-0.20
-0.40
-0.60
-0.80
-1.00
-1.20
-1.40
-1.60
-1.80
-2.00
0.00
20.00
40.00
60.00
80.00

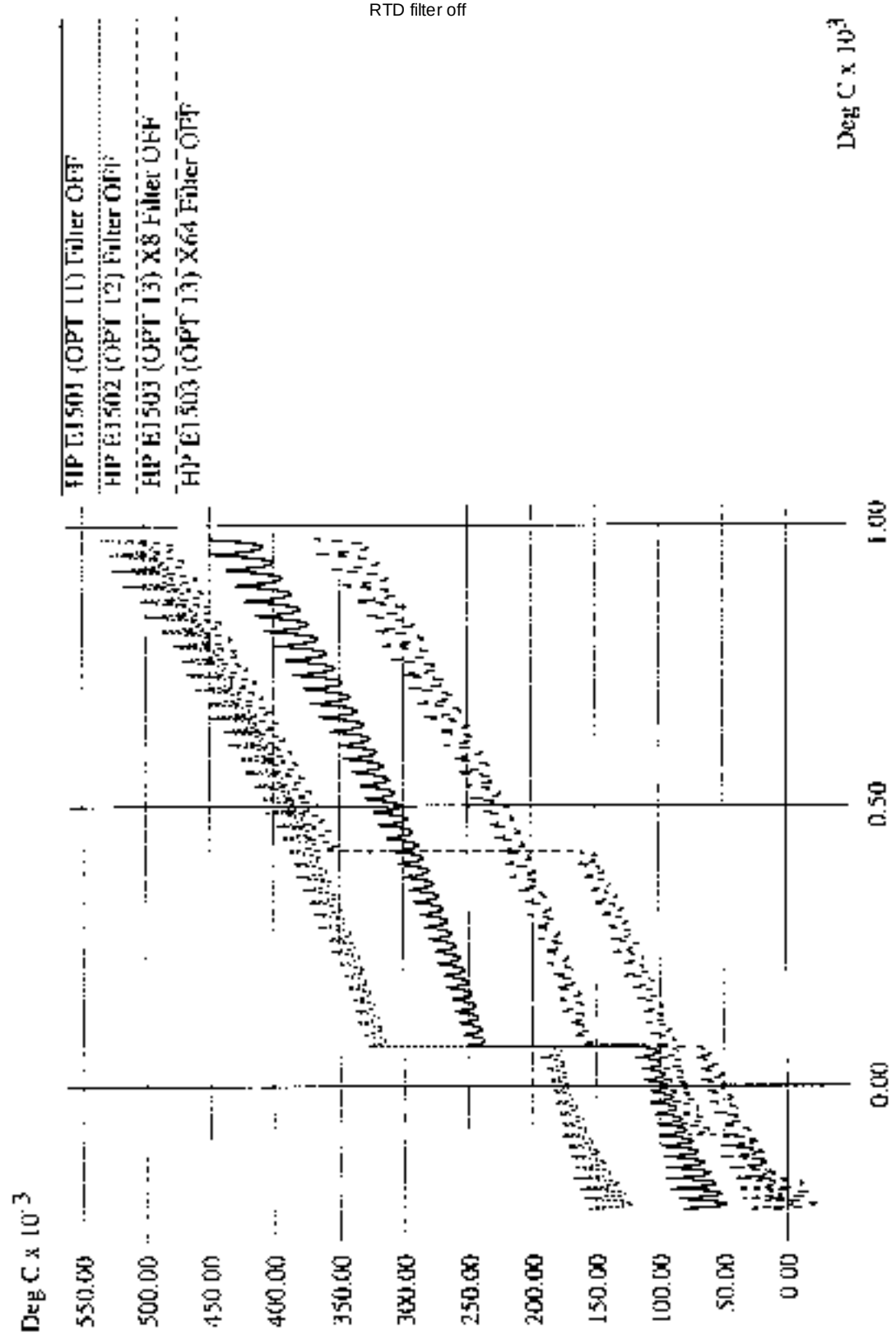
Reference Thermistor 5KΩ filter off (HP E1508/09)



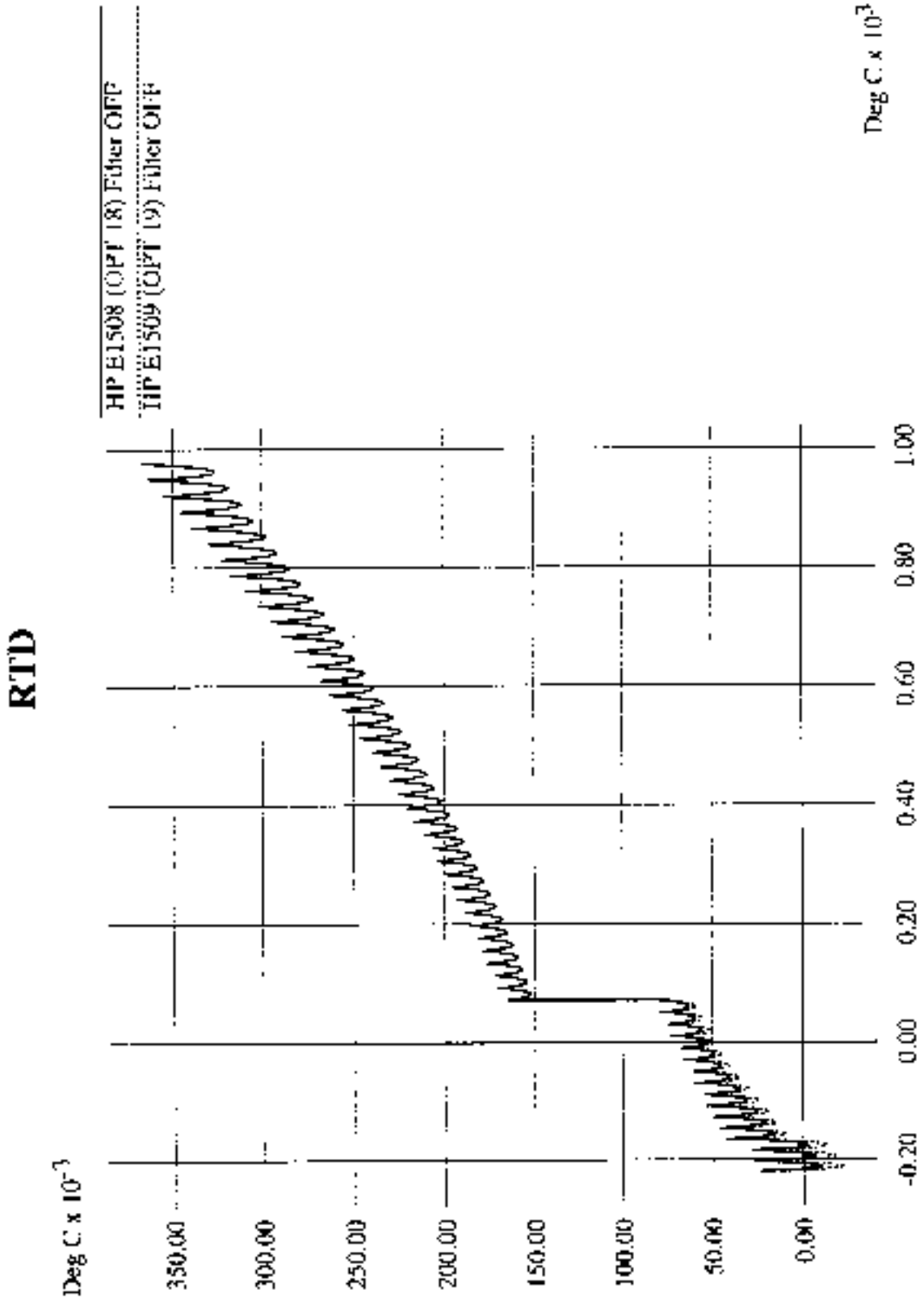
Reference RTD filter off

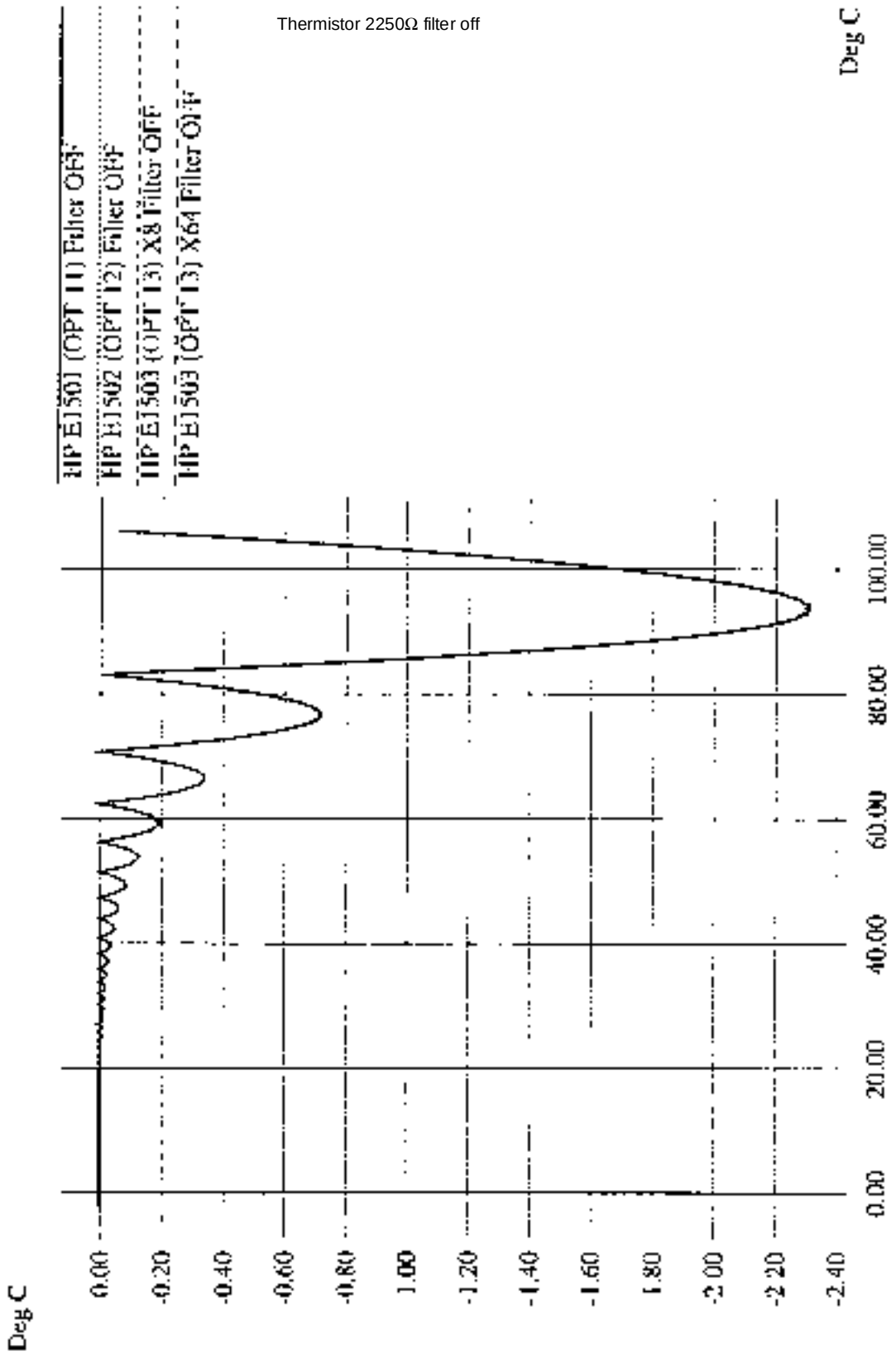


RTD filter off

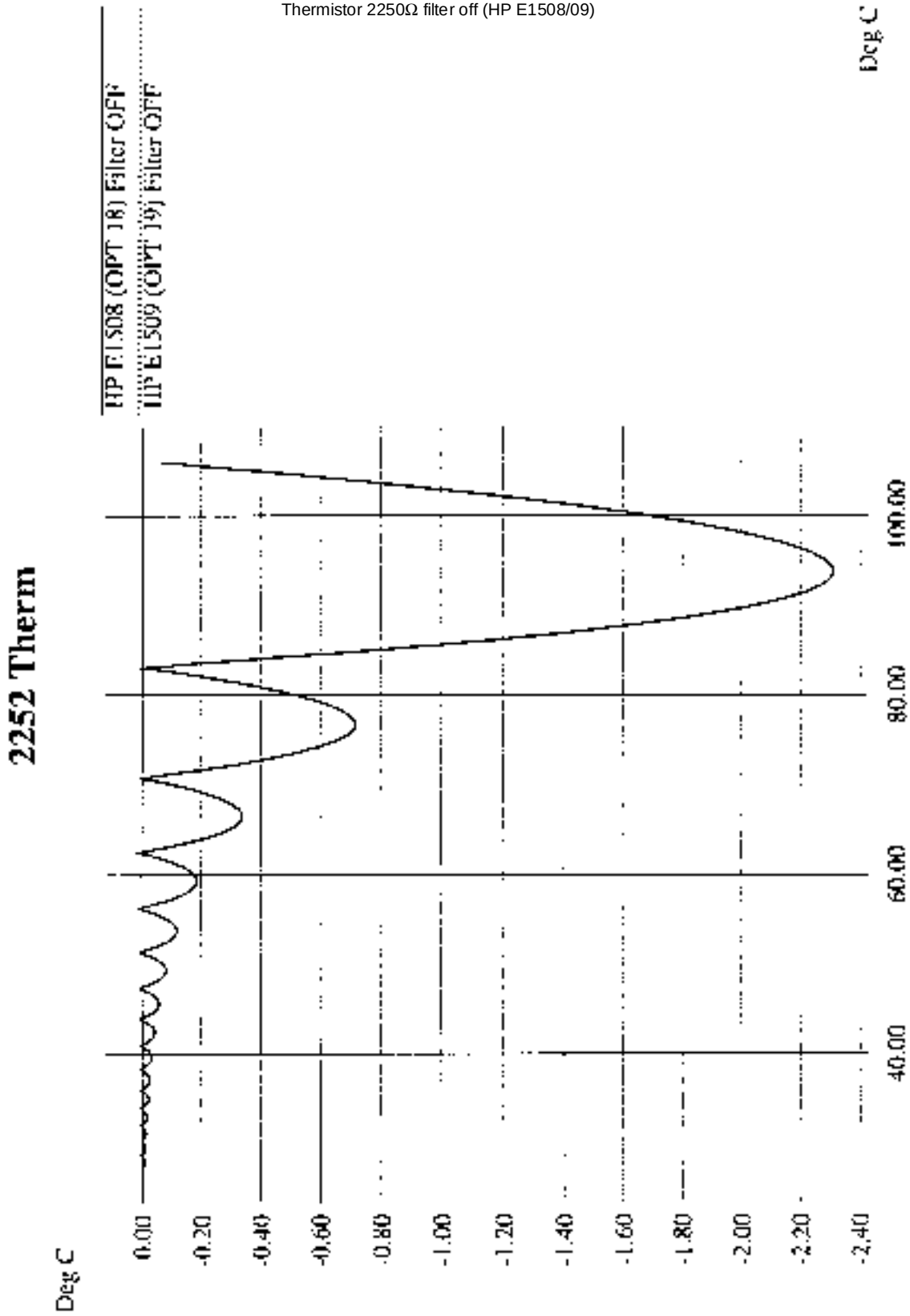


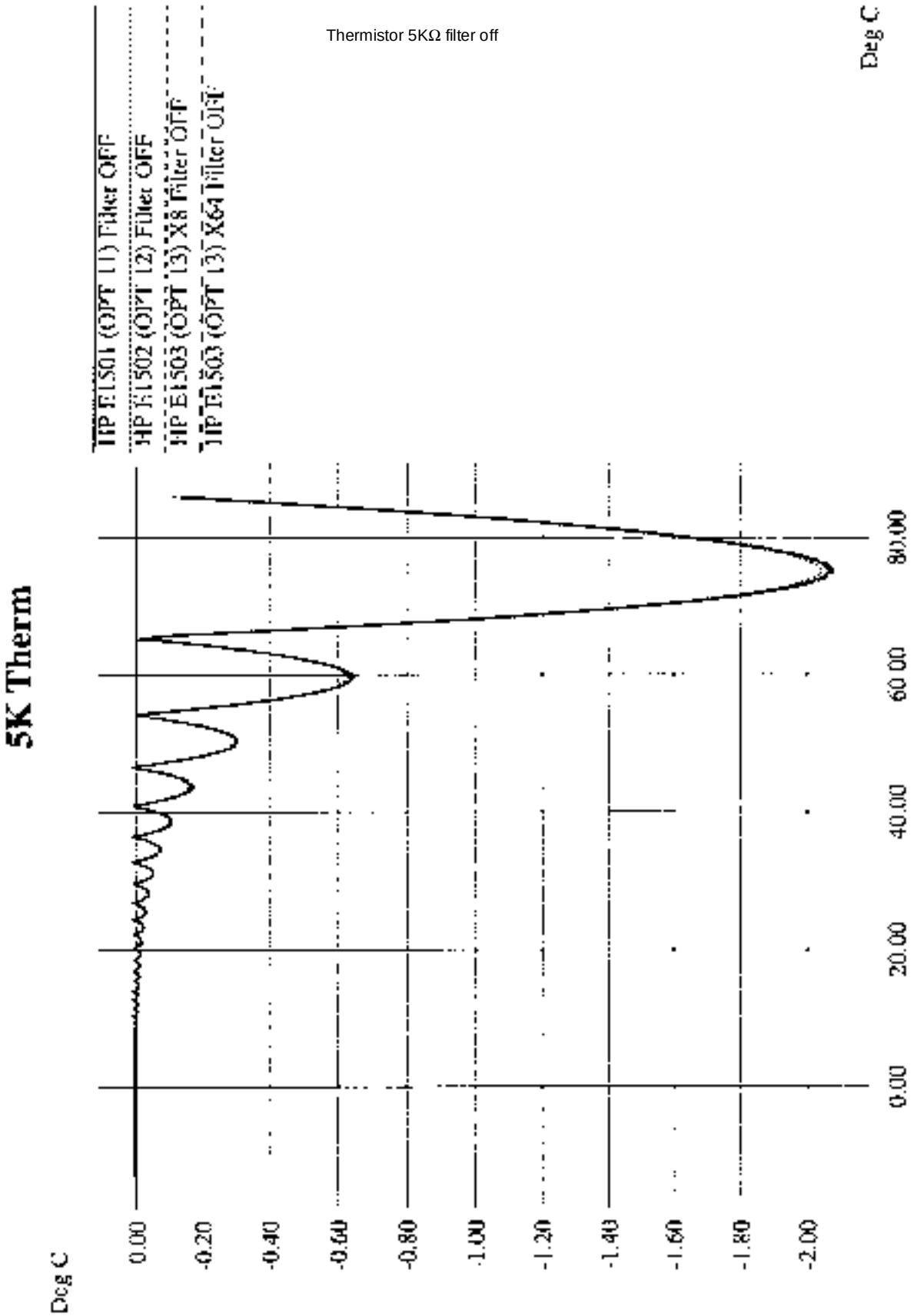
RTD filter off (HP E1508/09)



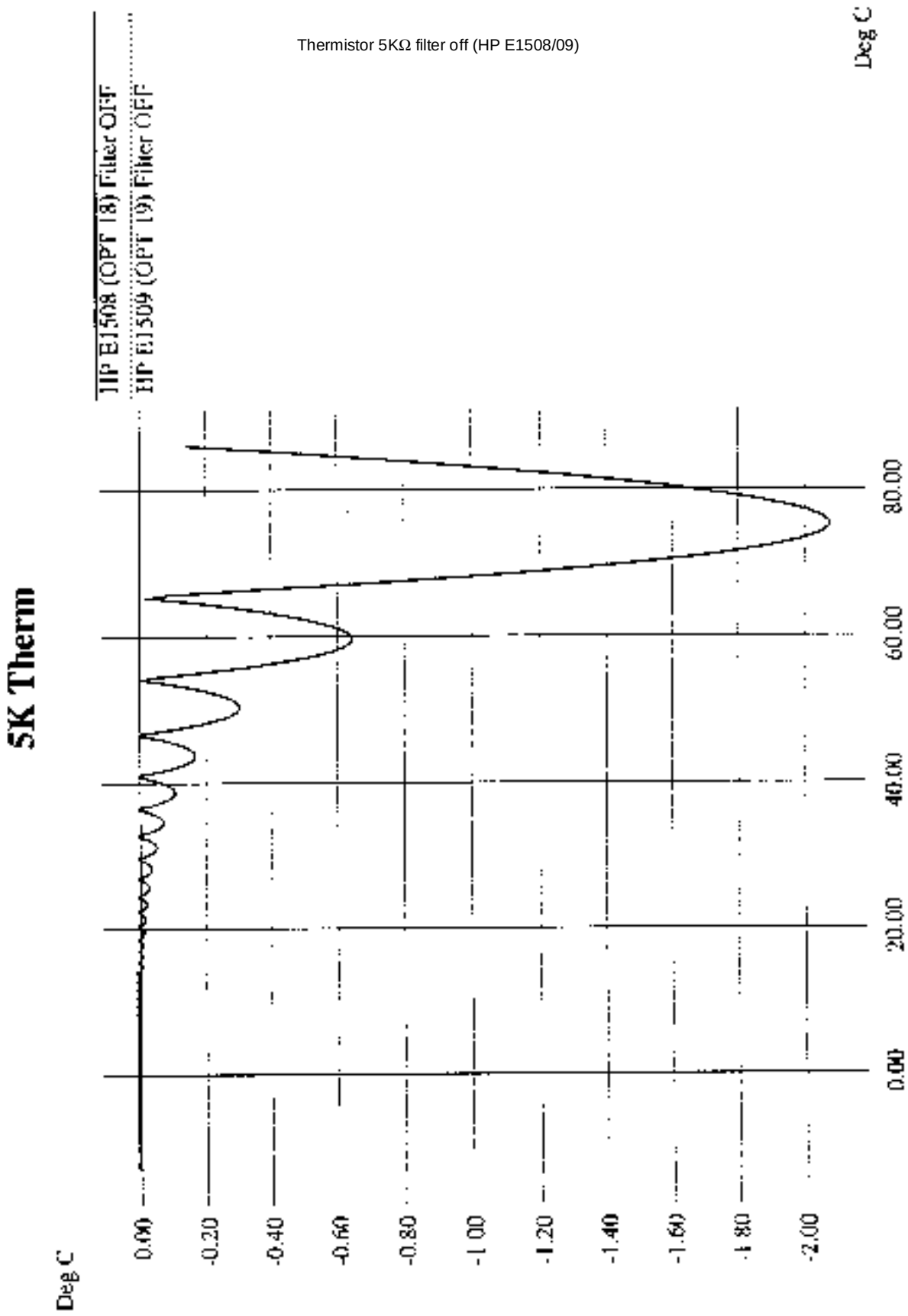

2252 Therm
Thermistor 2250Ω filter off
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C
Deg C
0.00
-0.20
-0.40
-0.60
-0.80
-1.00
-1.20
-1.40
-1.60
-1.80
-2.00
-2.20
-2.40
0.00
20.00
40.00
60.00
80.00
100.00

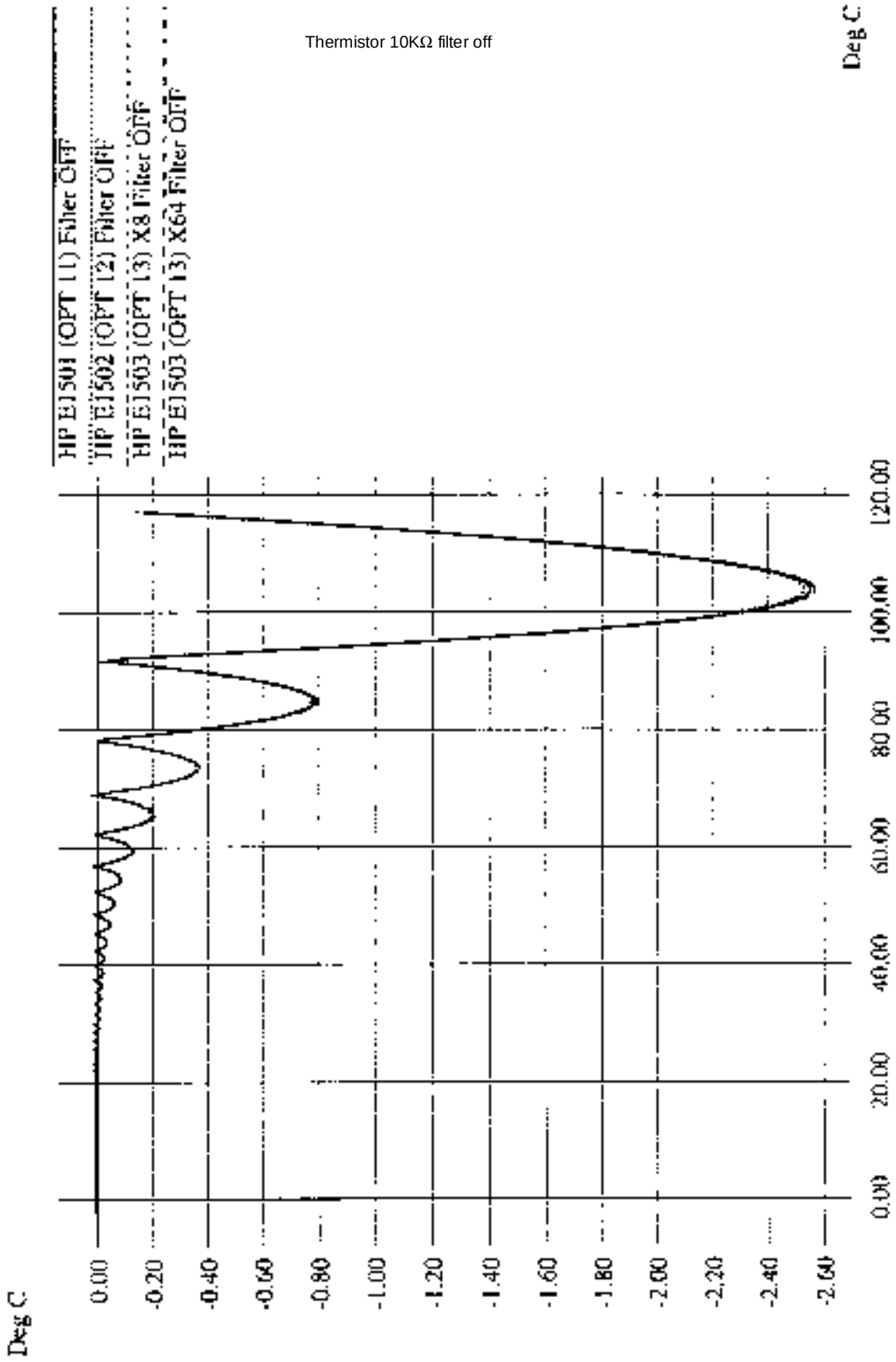
Thermistor 2250Ω filter off (HP E1508/09)



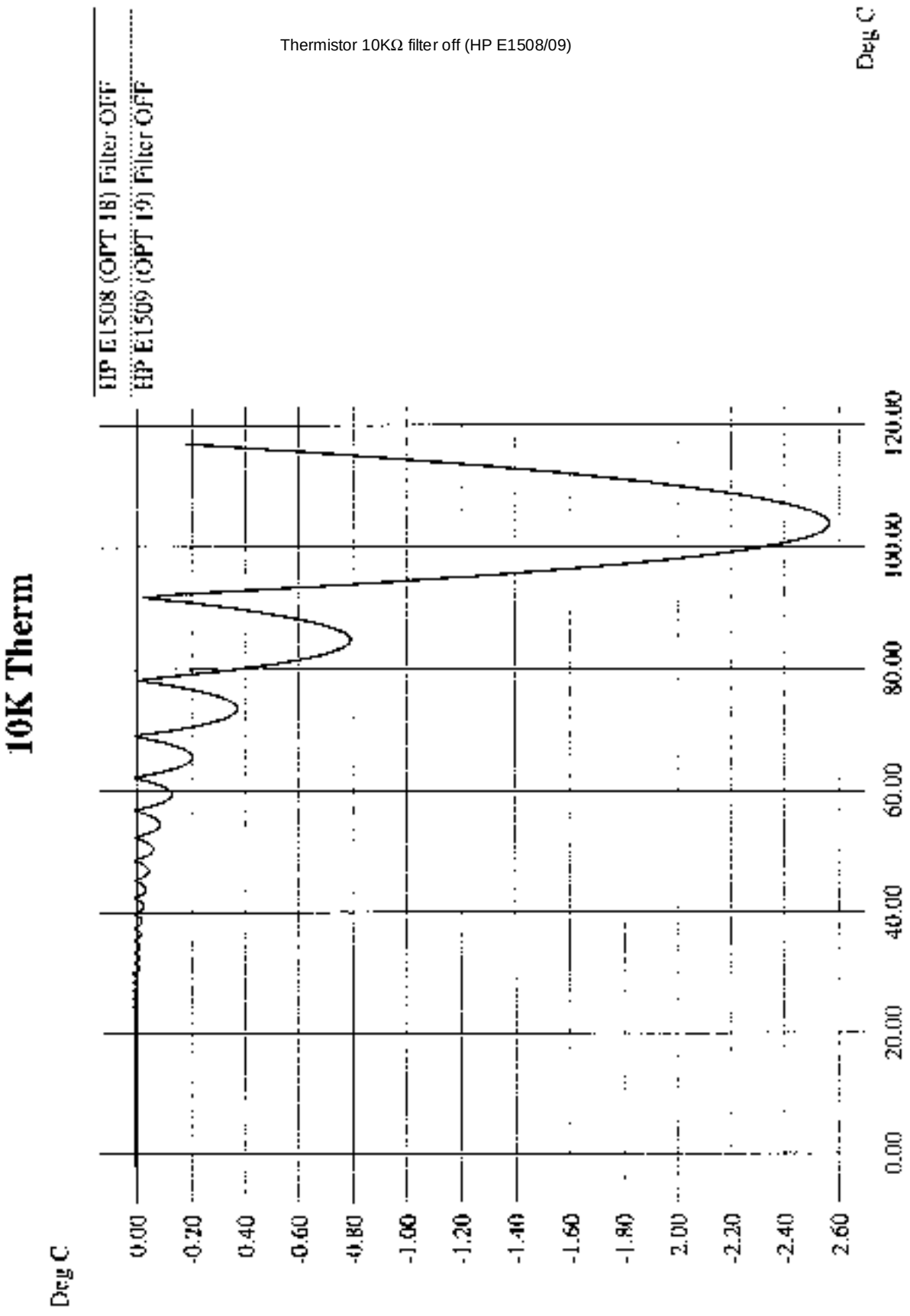

5K Therm
Thermistor 5KΩ filter off
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C
0.00
-0.20
-0.40
-0.60
-0.80
-1.00
-1.20
-1.40
-1.60
-1.80
-2.00
0.00
20.00
40.00
60.00
80.00
Deg C

Thermistor 5KΩ filter off (HP E1508/09)




10K Therm
Thermistor 10KΩ filter off
HP E1501 (OPT 11) Filter OFF
HP E1502 (OPT 12) Filter OFF
HP E1503 (OPT 13) X8 Filter OFF
HP E1503 (OPT 13) X64 Filter OFF
Deg C
0.00
-0.20
-0.40
-0.60
-0.80
-1.00
-1.20
-1.40
-1.60
-1.80
-2.00
-2.20
-2.40
-2.60
0.00
20.00
40.00
60.00
80.00
100.00
120.00
Deg C

Thermistor 10KΩ filter off (HP E1508/09)



Notes

Appendix B
Error Messages

Possible Error Messages:

-101	’Invalid character’.
-102	’Syntax error’.
-103	’Invalid separator’.
-104	’Data type error’.
-108	’Parameter not allowed’.
-109	’Missing parameter’.
-113	’Undefined header’.
-123	’Numeric overflow’.
-131	’Invalid suffix’.
-138	’Suffix not allowed’.
-141	’Invalid character data’.
-148	’Character data not allowed’.
-151	’Invalid string data’.
-158	’String data 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’.
-350	’Too many errors’.
-410	’Query INTERRUPTED’.
-420	’Query unterminated’.
-430	’Query deadlocked’.
+1500	’External trigger source already allocated’.
+2000	’Invalid card number’.
+2001	’Invalid channel number’.
+2002	’Invalid logical address’.
+2003	’Invalid word address’.
+2005	’No card at logical 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’.
+2101	’Failed Device’.
+2103	’Config warning, Device driver not found’.
+2105	’Config error 5, A24 memory overflow’.
+2108	’Config error 8, Inaccessible A24 memory’.
+2110	’Config error 10, Insufficient system memory’.
+2111	’Config error 11, Invalid instrument address’.

+2129	’Config warning, Sysfail detected’.
+3000	’Illegal while initiated’. Operation must be performed before INIT or INIT:CONT ON.
+3001	’Illegal while continuous’. While INIT:CONT ON and TRIG:SOUR IMM.
+3004	’Illegal command. CAL:CONF:RES or CAL:CONF:VOLT not sent’. Incorrect sequence of calibration commands. Send CAL:CONF:RES or CAL:CONF:VOLT command before CAL:VAL:RES or CAL:VAL:VOLT 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 contain at least two channels.
+3012	’Trigger too fast’. Scan list not completed before another trigger event occurs.
+3015	’Channel modifier not permitted here’.
+3019	’TRIG:TIM interval too small for SAMP:TIM interval and scan list size’. TRIG:TIM interval must allow for completion of entire scan list at currently set SAMP:TIM interval. See TRIG:TIM in Chapter 5, the Command Reference
+3020	’Input overvoltage’. Calibration relays opened (if JM2202 not cut) to protect module inputs, and Questionable Data Status bit 11 set. Execute *RST to close relays and/or reset status bit.
+3021	’FIFO overflow’. Lets you know that the FIFO buffer has filled and that one or more readings have been lost. Usually caused by continuous measurements taken faster than FIFO was read.
+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. See the [SENSe:]DATA:FIFO[:ALL]? command. Results in ASCII format, see the FORMat[:DATA] command.
+3037	’0x20: DSP-Invalid *CAL? constant or checksum. *CAL? required.’.
+3038	’0x40: DSP-Could not cal some channels’. Check that SCP is seated or replace SCP. Channel numbers are in FIFO. See the [SENSe:]DATA:FIFO[:ALL]? command. Results in ASCII format, see the FORMat[:DATA] command.
+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 CALibration:CONFigure:VOLTage.

+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'. Use SENSe:DATA:FIFO:ALL? to read the test results. For example, SENSe:DATA:FIFO:ALL? returns the values 72 and 108. This indicates that test number 72 failed on channel 8. NOTE: *TST? always returns results to the FIFO in ASCII format.

FIFO Value	Definition
1 - 99	ID number of failed test. See the following text for information on self-test error numbers.
100 - 163	Channel number(s) associated with field test (channel 0 - 63).
164	Main card A/D associated with failed test.
200	A/D range 0.0625 V associated with failed test.
201	A/D range 0.25 V associated with failed test.
202	A/D range 1.0 V associated with failed test.
203	A/D range 4.0 V associated with failed test.
204	A/D range 16.0 V associated with failed test.

Error ID Number	Corrective Action
1-19, 21-29	HP Service is required.
20, 30-37	Remove all SCPs and see if *TST? passes. If so, sequentially replace each SCP while running *TST? to identify failing SCP(s).
38 - 71	HP Service is required.
72, 74-76, 80-93	Try to re-seat the SCP associated with the *TST? failure, or move the SCP to a different SCP socket and see if the *TST? errors follow the SCP. If the test error follows the SCP, replace the SCP.
73, 77-79, 94-99	HP Service is required.

These are the only tests where the user should troubleshoot a problem. Other tests which fail must be handled by an HP repair facility. Make sure you record the values found in the FIFO for HP Service.

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

Notes

1. HP E1413/E1313 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. 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 higher than channel's CAL:LIM:UPP:DATA
+3061	'Cannot 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'
+3072	'Autorange not allowed with SENSE:FILTER on'. This filter is the A/D filter. Use of SCP filtering (filtering at channels) allows autoranging at high speed.
+3073	'Scan rate too fast with SENSE:FILTER on'. Minimum SAMPle:TIMer interval is 145 µsec while A/D filter is on. Use of SCP filtering (filtering at channels) allows sampling at maximum speed (SAMP:TIM 10us).

Notes

Appendix C
Glossary

Except where noted, all references to the HP E1413 apply to the HP E1313. The following terms are either unique to Hewlett-Packard's VXIbus modules or are unique to the HP E1413.

Automatic Scan List Sequencing	The HP E1413 can execute a user defined sequence of up to four Scan Lists, each of which can defining a sequence of up to 1024 channels to scan.
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 INITiate: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
List-of-Lists	Automatic Scan List Sequencing. The command ROUT:SCAN LISTL causes the HP E1413 to execute a sequence of Scan Lists defined in the special Scan List LISTL.
Scan List	A user defined list of up to 1024 channel specifiers that will be scanned when to module is triggered.
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

Except where noted, all references to the HP E1413 apply to the HP E1313. 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.

Note HP has a C language function library for the HP E1413 that provides virtually the same functionality as its SCPI command set. The library is provided as C language source code that you can modify as necessary and compile to run on your controller. Documentation and example files are provided with the library. Using these library functions is by far the fastest and most productive way to "register program" the HP E1413. Contact your HP sales representative for more information.

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

Table of Registers

Category	Address	Read Registers (return)	Write Registers	See Page...
Required VXI Registers	Base + 00_{16}	ID Register		338
	Base + 02_{16}	Device Type		339
	Base + 04_{16}	VXI Status Register	VXI Control Register	339 - 340
	Base + 06_{16}	Offset Register		341
Register-Based Command and Response Registers	Base + 08_{16}	Query Response Register	Command Register	341 - 342
	Base + $0A_{16}$		Parameter Register #1	342
	Base + $0C_{16}$		Parameter Register #2	342
	Base + $0E_{16}$		Parameter Register #3	342
Scan & Card Control Registers	Base + 10_{16}	Scan Status and Control Register		343 - 344
	Base + 12_{16}	Card Control Register		345
Interrupt Registers	Base + 14_{16}	Interrupt Configuration Register		347
	Base + 16_{16}	Interrupt Status Register		347
Virtual Instrument Registers	Base + $1A_{16}$	Common Capabilities Register		349
	Base + $1C_{16}$	Description Register		350
	Base + $1E_{16}$	Subclass Register		351
FIFO Registers	Base + 20_{16}	FIFO MSW		351
	Base + 22_{16}	FIFO LSW		351
	Base + 24_{16}	FIFO Status Register		352
	Base + $2A_{16}$	FIFO Reading Count Register		353
Trigger Registers	Base + 26_{16}		Software Trigger Register	353
	Base + $2C_{16}$	Trigger Timer Register		353
	Base + $2E_{16}$	Trigger Mode Register		354

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}=$C600_{16}$

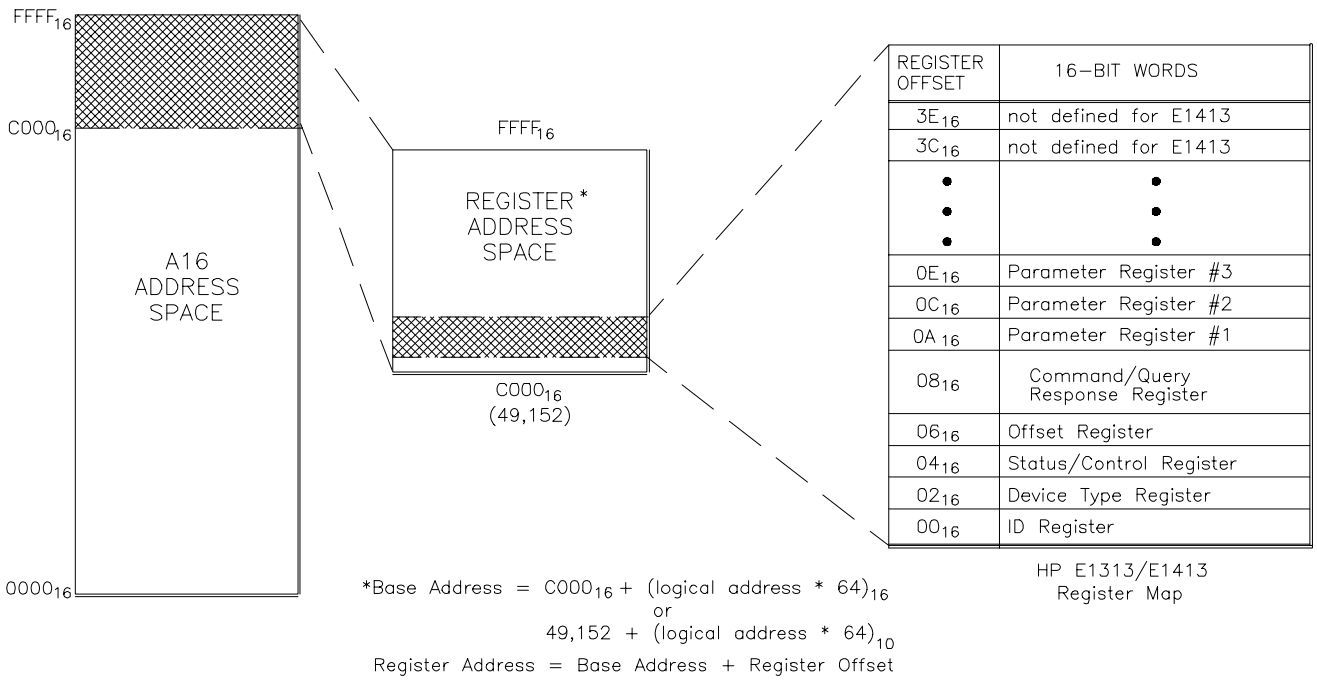


Figure D-1. HP E1313/E1413 Registers within A16 Address Space

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.

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		Manufacturer 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 E1313/E1413 is an extended device.

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 E1313/E1413 uses 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".

Device Type Register — Base + 02_{16}

Read only, returns $51C4_{16}$: The Device Type Register contains the required memory and model code.

Address	15 - 12	11 - 0
Base + 02_{16}	Required Memory Code	Model Code

Required Memory: The HP E1313/E1413 supports 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 $1C4_{16}$.

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	13-12	11-10	9-8	7	6	5-4	3	2	1	0
Base + 04_{16}	A24 Active	MODID*	not used	Channel Count (E1313)	not used	DONE	NOERR	not used	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.

Channel Count (HP E1313 only): 00_2 = 32 channels, 10_2 = 64 channels. Values 01_2 and 11_2 indicate invalid configurations. Read the Model Code in the Device Type Register first to differentiate between HP E1313/E1413. All HP E1413s have 64 channels.

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.

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.

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'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	not used

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 10 least significant bits are all 0.

Register-Based Command and Response Registers

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 356.

Address	15 - 0
Base + 08_{16}	Query Response

Some query commands may return multiple query responses.

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 356.

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 de-asserted (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 and 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.

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 de-assert 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 is 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 381.

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.

Auto-arm: 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 Auto-arm 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 will not 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.

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 de-asserted) 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 de-asserted after the trigger is recognized by the control processor. In Auto-arm mode, Armed toggles.

Initiated: When asserted (1) this bit indicates that the module is initiated. Initiated is de-asserted 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 de-asserted, but Armed is asserted (1), this field shows the scan list which will be executed upon receipt of a trigger. These bits are undefined when executing a List-of-Lists scan.

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}	not used	FIFO Mode	not used	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 on-board 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 JM2201 must be set to the Enable position.

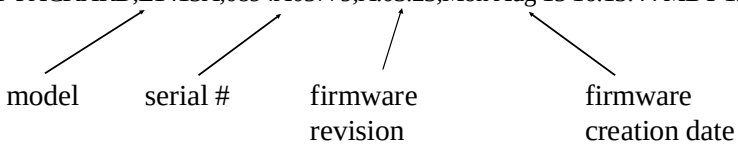
A24 Window: These three bits allow A24 access to various 256 Kbyte windows of the control processor's memory. The following table and Figure D-2 show the relationship between bits 10 - 8 and the memory pages mapped.

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.
1	1	0	Flash Memory (writes require jumper JM2201 to enabled position).
1	1	1	A/D RAM and DSP I/O Space.

Example: Retrieving the Firmware ID String

An example of accessing module memory through the A24 Window is retrieving the module ID information. Set the A24 Window bits to 110_2 to point to the module's Flash Memory. Now, at Base+80_{16} begins a string of characters that form the module's ID string. An example of this string is:

HEWLETT-PACKARD,E1413A,0894A05779,A.03.23,Mon Aug 15 16:13:44 MDT 1994



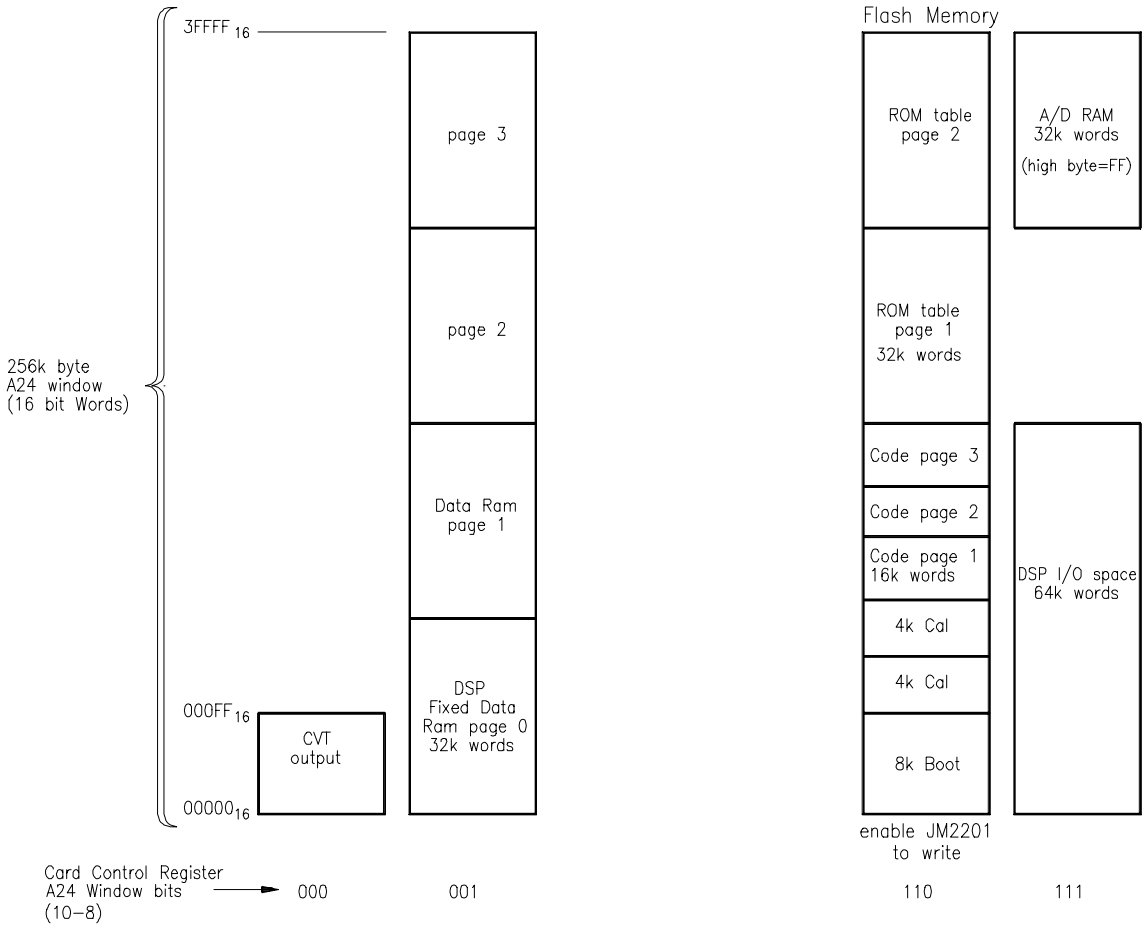


Figure D-2 A24 Memory Mapping

Notes

1. Normally, bits 10 - 8 are zero (0), so that the Current Value Table can be accessed through A24 address space.

2. CVT channel values start at Base + 00000_{16} (channel 0) with each channel location incremented by 4 (byte addressing).

   $$CVT\ Channel\text{–}to\text{–}address = Base + 00000_{16} + channel\_number \times 4$$

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	Meas Complete	Trig Too Fast	Over V Detect	Scan Complete	FIFO over-flowed	FIFO Half full	not used	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

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	Meas Complete	Trig Too Fast	Over V Detect	Scan Complete	FIFO Over-flowed	FIFO Half Full	not used

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 de-asserted.

Limit Test Exceeded: 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: 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.

Measurement Complete: A one (1) indicates that the measurement 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 E1313/E1413 modules have.

Common Capabilities Register — Base + $1A_{16}$

Read only, returns 50_{16}: This register indicates whether the HP E1413 supports 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).

Description Register — Base + $1C_{16}$

Read only, returns $FA60_{16}$: The Description Register returns more information on identification and functionality of the HP E1413. 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 the following:

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.

Subclass Register — Base + $1E_{16}$

Read only, returns $7FFF_{16}$: The Subclass Register identifies the HP E1413 as an HP virtual instrument device.

FIFO Registers

The FIFO registers consist of the FIFO MSW, FIFO LSW, the FIFO Status, and FIFO Reading Count Registers.

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. MSW Registers and LSW Registers combine to become a 32-bit Motorola format floating point value.

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 de-asserted, a bus error will occur.

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, 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.

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 = $(1.0E{-}4{*}reg\_value){+}1.0E{-}4$

Address	15 - 0
Base + $2C_{16}$	trigger interval (0 - 65535)

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 + $2E_{16}$	Trigger Enable	undefined (write 0)	Output Trigger Source	Trig Timer Mode	Select Trigger Timer	Trig Out	Trig Mode	Trig/Arm Source

Trigger Enable: Set to one (1) to enable triggers. Always set to zero before changing other bits in this register, then set back to one.

Output Trigger Source: Selects the source driving the TTLTRG*n* line when the Trig Mode is "Output".

Bit 12	Bit 11	Output Trigger Source
0	0	"Trigger In" Signal
0	1	First Trigger (see FTRigger on page 217)
1	0	Limit Test Exceeded
1	1	SCP Trigger

Trig Timer Mode: 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). For more information see TRIGger:TIMer:MODE on page 270.

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 example 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 (overwritten) by the next query command. Always read query responses before executing the next query command. For example the value 9.9E+37 is $7E94F56A_{16}$, $7E94_{16}$ is sent to the lower numbered parameter register, while $F56A_{16}$ 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 categories listed on the following page.

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-Could not 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 cannot 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.

LOW_LIMIT <*channel*> <*highword*> <*lowword*> 0030_{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.

UP_LIMIT <*channel*> <*highword*> <*lowword*> 0040_{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.

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 a 16-bit value, the lower 4-bits of which are significant. 0000_2 indicates an HP E1413A, 0001_2 indicates an HP E1413B, and 0010_2 indicates an HP E1413C. This command is provided to allow your custom application to determine if you can access the extended hardware capability of the B or C version. The B and C version can respond to VXIbus D32 data transfers while the A version is a D16 device. The VXIbus resource manager in your system will automatically use the proper data transfer mode.

SCBREAD? <*regaddr*> 0800_{16}

Returns a 16-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	

Plug-on#:	SCP slot number (0 - 7)
Chan_addr:	Channel number on SCP () - 7)
Reg_addr:	SCP register address (see individual SCP reference)

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 16-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 = $00ppp000000_2$)
(ppp = SCP plug-on number, 0 - 7)

15 - 8	7 - 0
ID	SUB ID

Sub ID is an 8-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 8-bit ID field determines the class of SCP, which directs the SCPI driver how to control it. Simple SCPs may return the same number in both fields.

Scale Register (Read/Write, regaddr=00ppp000001$_2$)
(ppp = SCP plug-on number, 0 - 7)

15 - 4	3	2 - 0
do not care	Sign	Scale

Only the low four bits of this register are significant. They consist of a sign bit and a 3-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. (See individual SCP Reference Manuals for details). The SCALE factor is used in conjunction with each channel's Gain Register value to assess the actual gail for each channel. If SCALE is a positive value, then the gain is 1 or greater; if SCALE is negative, then the gain is 0-1.

Channel Registers (CHN bit = 1)

Gain Register (Read/Write, regaddr=01pppccc001$_2$)
(ppp = SCP plug-on number, 0 - 7. ccc = SCP channel number 0 - 7)

15 - 4	3 - 0
do not care	Gain

Only the low 4-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. See the individual SCP Reference Manual for details.

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.

The actual channel gain may then be computed as:

$$GAIN = 2^{SHIFT}$$

Example 1: If an SCP module has SCALE = 0010b, and CHGAIN programmable, then when CHGAIN = 0000, the channel has unity gain (SHIFT = 0). When CHGAIN = 0001b, the channel has a gain of x4 (SHIFT = 2). When CHGAIN = 0010b, the channel has a gain of x16, and so forth.

Example 2: If 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: If an SCP module has CHGAIN=0001b (fixed) for all channels; and if the bottom 3-bits of SCALE are programmable with the sign bit is fixed at 1. Then the SCP is a programmable attenuator (gain <1.0), 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 HP E1413 does not support attenuation by greater than 2 (on standard SCP modules) due to fixed-point arithmetic limits. This does not prevent SCPs from implementing larger attenuation factors, but such factors will not 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. <*range*> is an integer from -1 to 4, where 4 indicates the highest 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 the table on page 358).

READTEMP 2000_{16}

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 the on-card resistor is 7500.0 Ohms.

RESIST 4020_{16}

The Calibration Bus Relay and the Ohm Relay switched to allow 4-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*>. <*range*> is an integer from -1 to 4, where 4 indicates the highest range. 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. <*range*> is an integer from -1 to 4, where 4 indicates the highest range. 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. <*range*> is an integer from -1 to 4, where 4 indicates the highest range. 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}

If the Flash Memory Enable jumper (JM2201) is set to enable, the Channel Tare offset constants are copied into Flash Memory.

TAREAPPEND <*channel*> 3110_{16}

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}

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 the table on page 358).

TARENULL 3100_{16}

Clears the Tare Channel List used by the TARECAL command. Add channels to the list with TAREAPPEND.

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.

ADVRATE*n* <*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.

APPEND*n* <*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. Unless you choose autorange, you must set the range so the A/D can accommodate the expected SCP output from the specified channel. 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. Unless the A/D is overloaded (wrong A/D range for the SCP gain and input) or the input to the SCP is overloaded (generally greater than ±16 VDC) the measurement value returned always reflects the signal value at the input to the channel's SCP. The module's Control Processor accounts for the A/D range and SCP gain settings. The following table summarizes the data flag choices:

Data Flags (Bits 2 - 0)	Meaning
$00x_2$	No data destination (reference 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 = do not 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*> 0114_{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 for Standard EU Conversions

(See the next discussion for Custom EU Conversions). ASSIGN loads the specified channel's linear table area with the correct values for the indicated conversion. If <*conversion*> is from 1 through 1516, it indicates a standard linearization, such as an 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 standard 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} - $1F_{16}$	Reserved (not used)	

ASSIGN for Custom EU Conversions

The ASSIGN command is also used to access custom EU tables previously loaded into the specified channels' linearization table area. The <*compensation*> parameter is used only if the custom EU is a thermocouple conversion which needs reference compensation. For custom EU conversion, the bit positions of the <*conversion*> parameter take on the following meanings:

15 - 7	6	5	4	3 - 0
always 0	Custom	Reference	Piecewise	Shift Factor

Bit 6, Custom: determines whether the EU conversion is standard (as previously discussed), or custom. Set bit 6 to 1 for custom (40_{16}).

Bit 5, Reference: when set, specifies that this channel is a custom temperature conversion that is to be used as the thermocouple reference junction temperature for all subsequent thermocouple measurements.

Bit 4, Piece Wise: when 1, indicates that the linearization table is a piece wise conversion table. When 0, indicates the table is linear.

Bit 3-0, Shift Factor: specifies the Shift Factor based on the range the table was built on. This is the negative-to-positive voltage range that the custom conversion table covers. The Shift Factor versus voltage range is as follows:

Shift Factor	± Voltage Range
0_{16}	.015625
1_{16}	.03125
2_{16}	.0625
3_{16}	.125
4_{16}	.25
5_{16}	.5
6_{16}	1
7_{16}	2
8_{16}	4
9_{16}	8
A_{16}	16
B_{16}	32
C_{16}	64

The <*compensation*> parameter is used only for custom EU thermocouple measurements. <*compensation*> specifies which built-in (standard) table to use for reference junction compensation. <*compensation*> allows 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. Standard 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 piecewise 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. Contact your HP System Engineer for information on creating and downloading custom EU conversion tables.

5. All Piecewise EU Conversions have 16-bit dynamic range. Linear EU conversions have full dynamic range of the HP E1413.

6. Piecewise 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.

7. Linear EU Conversions use y=Mx+B, where x is the entire 16-bit A/D reading, M and B are constants.

NEW*n* 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 as follows:

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 run-time 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 $0C00_{16}$ in its Data Space RAM in place of the normal run-time 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. <*range*> is an integer from -1 to 4, where 4 indicates the highest range. 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

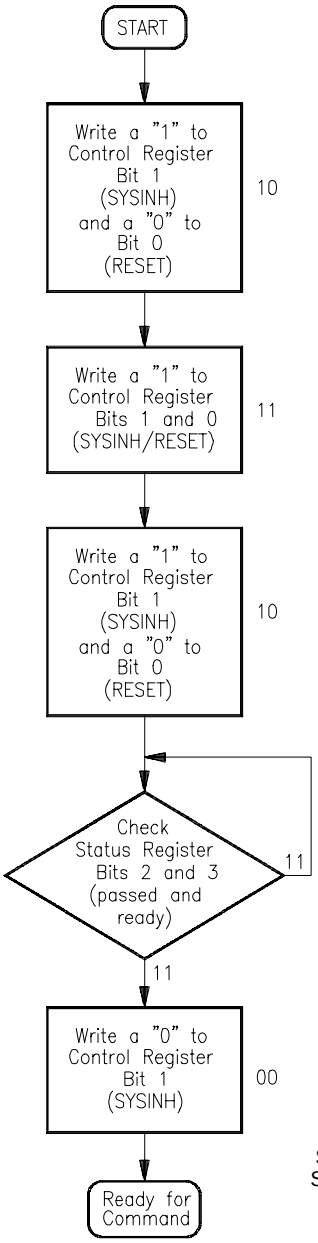
Register-based programming begins with a few fundamental register-based operations. They are:

- Resetting the module
- General register access
- Executing register based commands
- Querying the module

These operations are explained here in detail. In " Programming Sequence" on page 382, they will each be shown as a single operation.

Resetting the Module

Resetting the module places it in the Ready for Program State. The module is reset according to Figure D-3.



See the Control Processor State Diagram on page 381

Figure D-3. Resetting the HP E1313/E1413

Comments

- The registers used are:
  - Control Register (base+04_{16})(write)
  - Status Register (base+04_{16})(read)
- 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 the Control Processor State Diagram on page 381). 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.

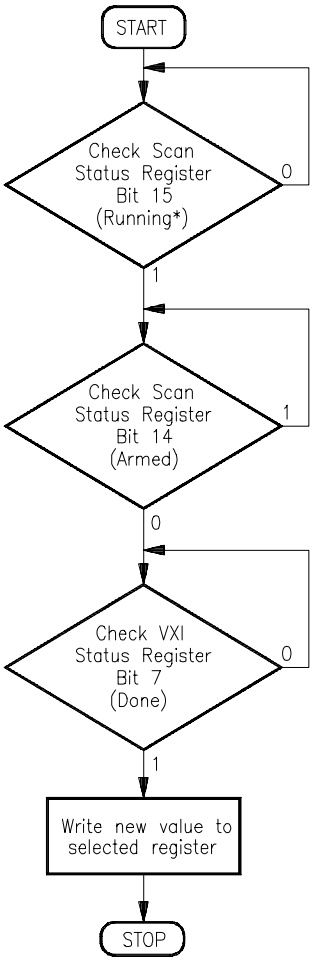


Figure D-4. General Register Access

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 Figure D-5.

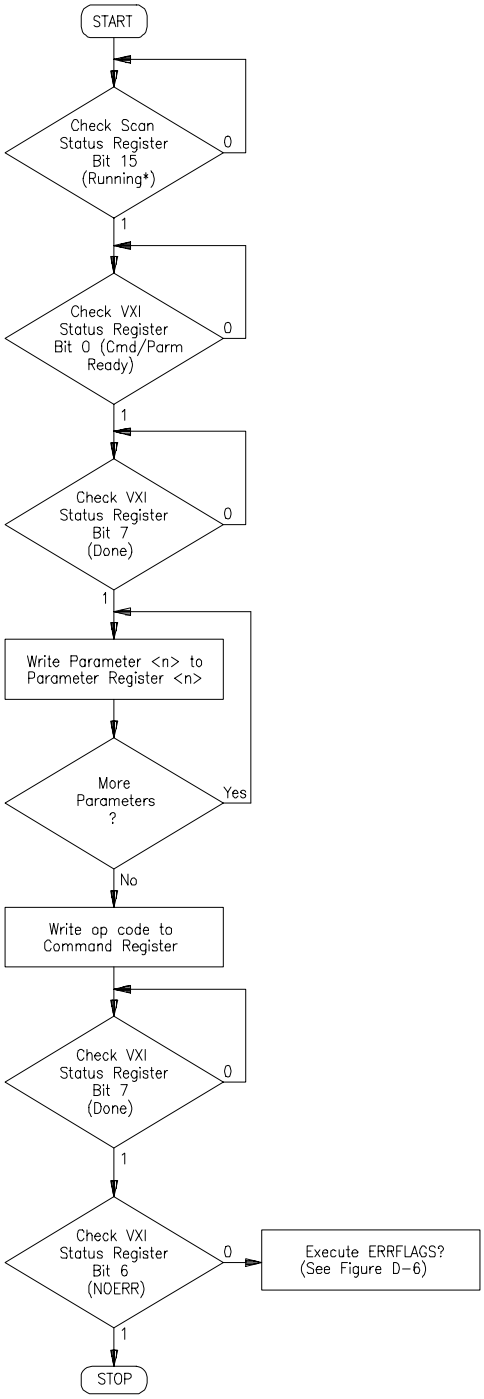


Figure D-5. Executing Commands

Comments

- 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 and Control Register (base+10_{16})
- The parameters must be written to their registers before the command is written. Execution begins as soon as the command is written.
- If the 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 shown in Figure D-6.

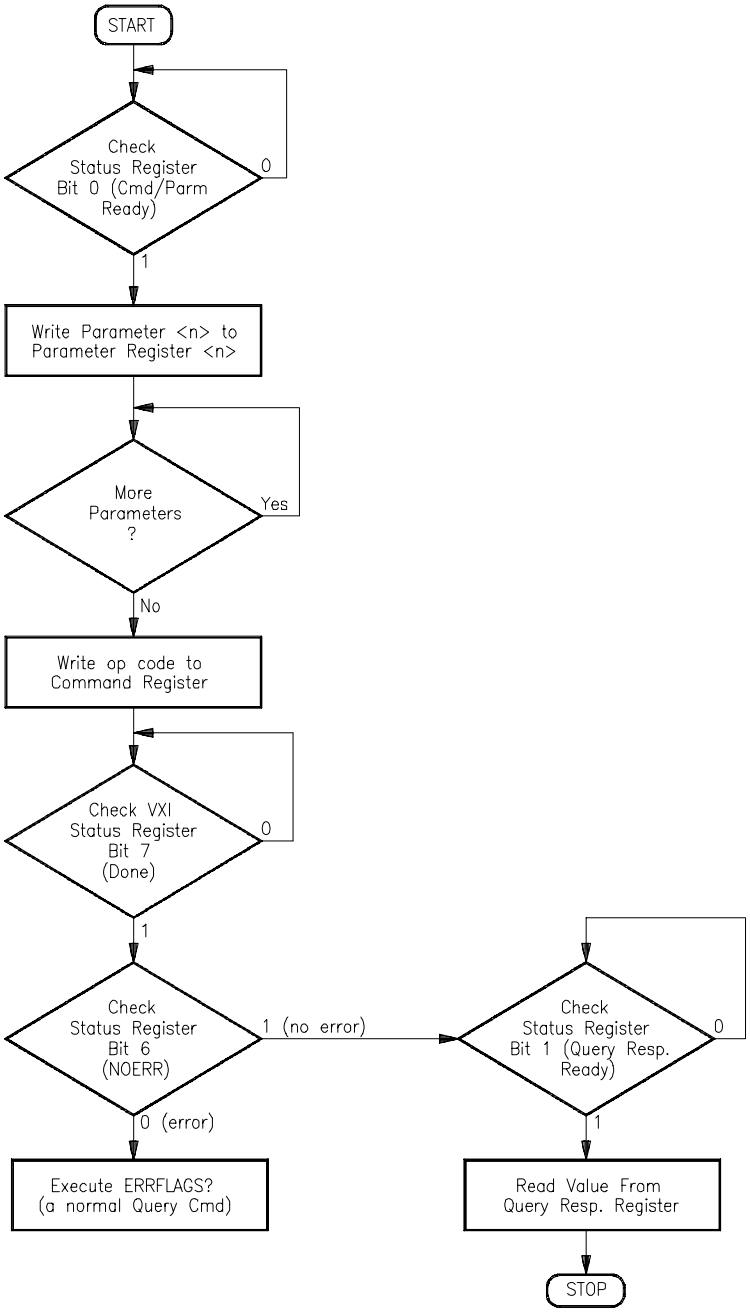


Figure D-6. Executing Queries

Comments

- 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})
- 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

Figure D-7 shows the relationships between the control processor's states.

Reset and **Abort** are bits written to the VXI Control Register (Base + 04_{16}).

Passed and **Ready** are bits read from the VXI Status Register (Base + 04_{16}).

Running* is a bit read from the Scan Status Register (Base + 10_{16}).

ARM is a Register-Based command sent to the Command and Parameter Registers (Base + 08_{16} through $0E_{16}$).

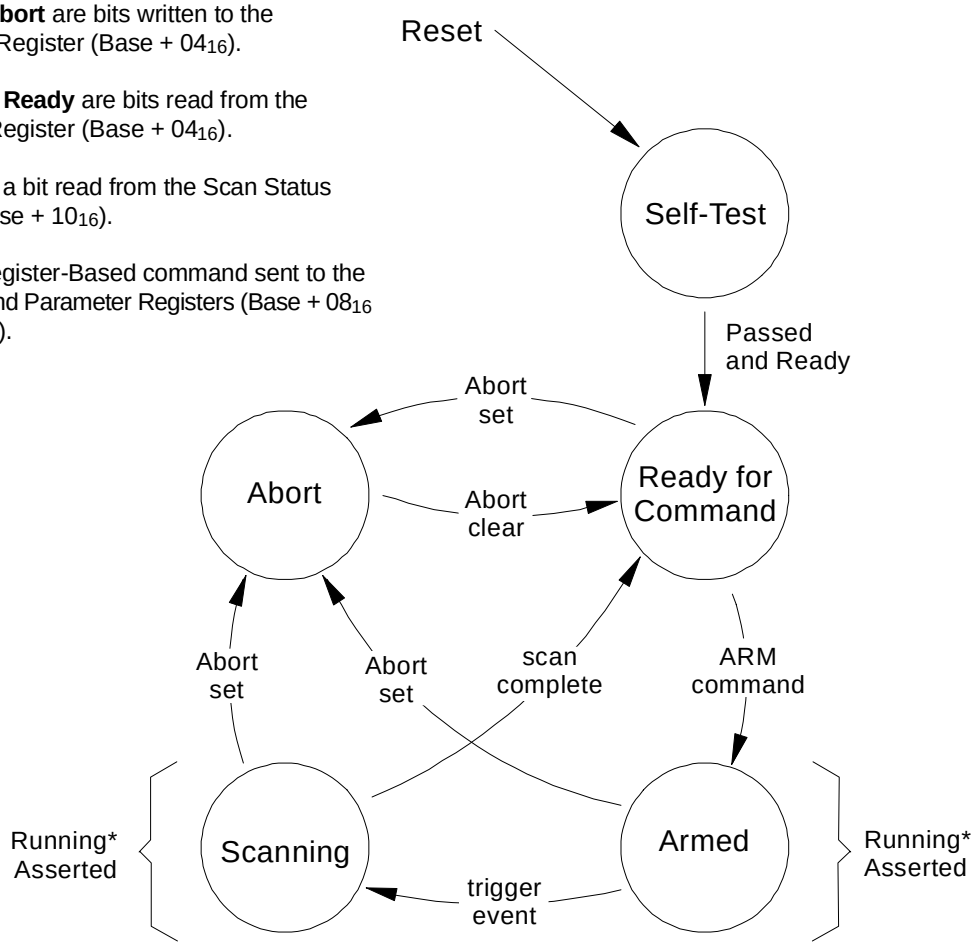


Figure D-7. Control Processor State Diagram

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 sections 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 be changed 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 and 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 page 372)
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

Calibrate SCP Channels

Reg Offset (hex)	Value (hex)
08	1000 (CARDCAL cmd) note 2

Clear Scan List 1

Reg Offset (hex)	Value (hex)
08	0100 (NEW*n* 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 and CVT)
08	0110 (APPEND cmd) notes 2,4

Select Scan List 1

Reg Offset (hex)	Value (hex)
10	0000 (See Scan Control Bits on page 343)

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

Appendix E
Using HP VEE with the HP E1313/E1413

General

The absolute fastest system performance is obtained with an embedded computer connected directly to the VXI backplane, running a real time operating system, and using all custom compiled C code. If this is not required for your application, or you do not need the absolute fastest performance, then use this appendix to program the HP E1313/E1413 using HP VEE. This appendix shows how to get good performance from HP VEE and the HP E1313/E1413 without writing any compiled C code.

The hardware in the HP E1313/E1413 is designed such that it can constantly transfer readings to a computer while it continues to take more readings. The rate at which these readings are transferred determines the rate at which data can be continuously acquired.

There are three main ways that HP VEE can communicate with the HP E1313/E1413.

Communication Method	Characteristic
Instrument panel	Easy to use, slow to execute
Direct I/O SCPI commands	Harder to use, medium speed
Direct I/O register access	Hardest to use, fast

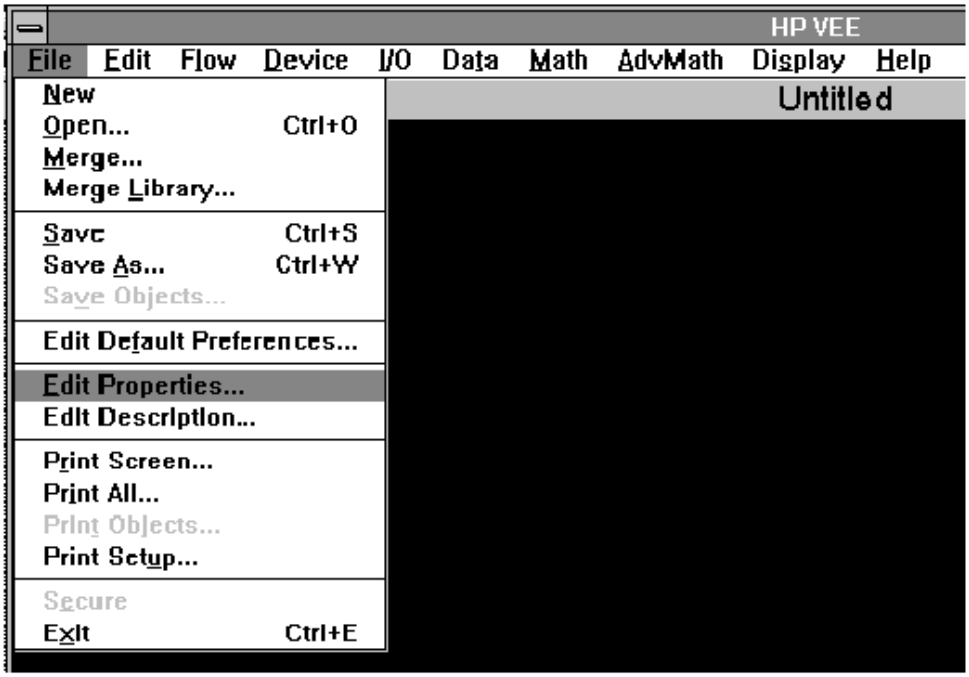
How to Use HP VEE with the HP E1313/E1413

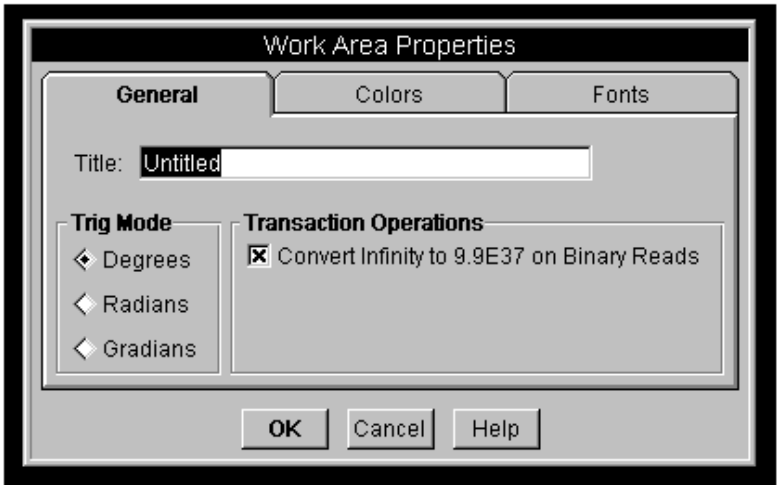
A good strategy to use with HP VEE and the HP E1313/E1413 is to use an instrument panel to send SCPI commands to configure the HP E1313/E1413 as this will be easy, and for the most part, execution speed does not matter. After the HP E1313/E1413 has been configured using an instrument panel, then use Direct I/O SCPI commands to start and monitor operations. Readings should be moved using Direct I/O register access in systems with a direct backplane connection or with Direct I/O SCPI commands, if a command module (HP E1406A, for example) is used. Moving data in the fastest possible manner is described as follows.

1. The HP E1313/E1413 has registers that contain data in 32-bit real format. This data can be used directly by HP VEE if one requirement is met. This requirement is to have HP VEE version 3.0 or later for UNIX® controllers or HP VEE version 3.2 for PCs. This version of HP VEE has a global parameter that must be configured.

In the HP VEE menu:

Files
    Edit Preferences
check Convert infinity to 9.9E37





This sets up HP VEE so it can directly access data from the HP E1313/E1413 FIFO and CVT registers. This is required because the HP E1313/E1413 has implemented IEEE 754 Floating Point Format which includes special codes for "plus infinity", "minus infinity", and "not a number". These special codes causes a floating point math error in HP VEE, thus they must be converted to a number which HP VEE can handle such as 9.9E37.

2. If you have a configuration that provides direct VXI backplane access (VXLINK, MXI, Embedded), you need to configure the HP VEE I/O to be able to directly access the HP E1313/E1413 FIFO register and the CVT table.

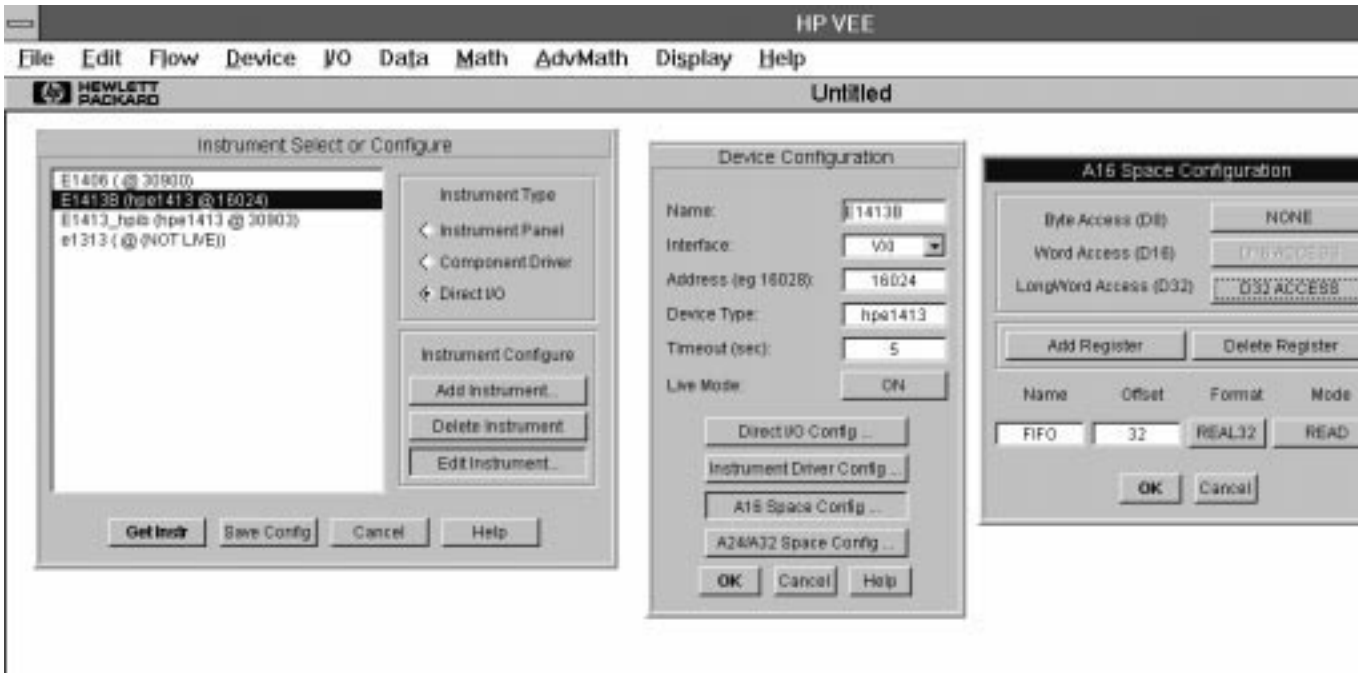
   Configuring to access the FIFO is done by selecting:

   I/O
   Instrument
   Highlight the HP E1313/E1413 in your system
   Edit instrument
   A16 Space Config…

   Then set the following fields:

Byte Access D16	NONE	
Word Access D16	D16 Access	always selected
Long Word Access (D32)	D32 Access	

Name	Offset	Format	Mode
FIFO	32	Real 32	Read



The FIFO register, which is the top of the 64K reading first in first out hardware, is located in A16 address space at offset 32. Each time this register is read, one reading is removed from the hardware FIFO.

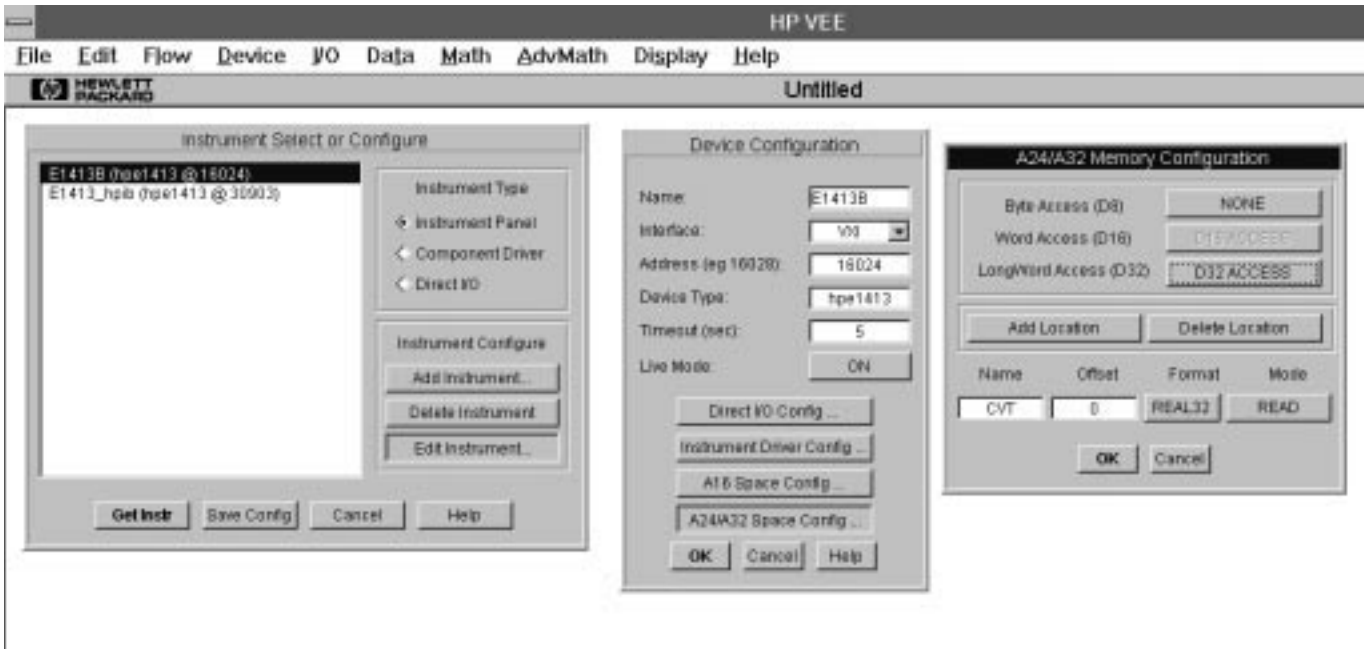
Direct access to the HP E1313/E1413 CVT table is configured by selecting:

I/O
Instrument
Highlight the HP E1313/E1413 in your system
Edit instrument
A24/A32 Space Config…

Then set the following fields:

Byte Access D16	NONE	
Word Access D16	D16 Access	always selected
Long Word Access (D32)	D32 Access	

Name	Offset	Format	Mode
CVT	0	Real 32	READ



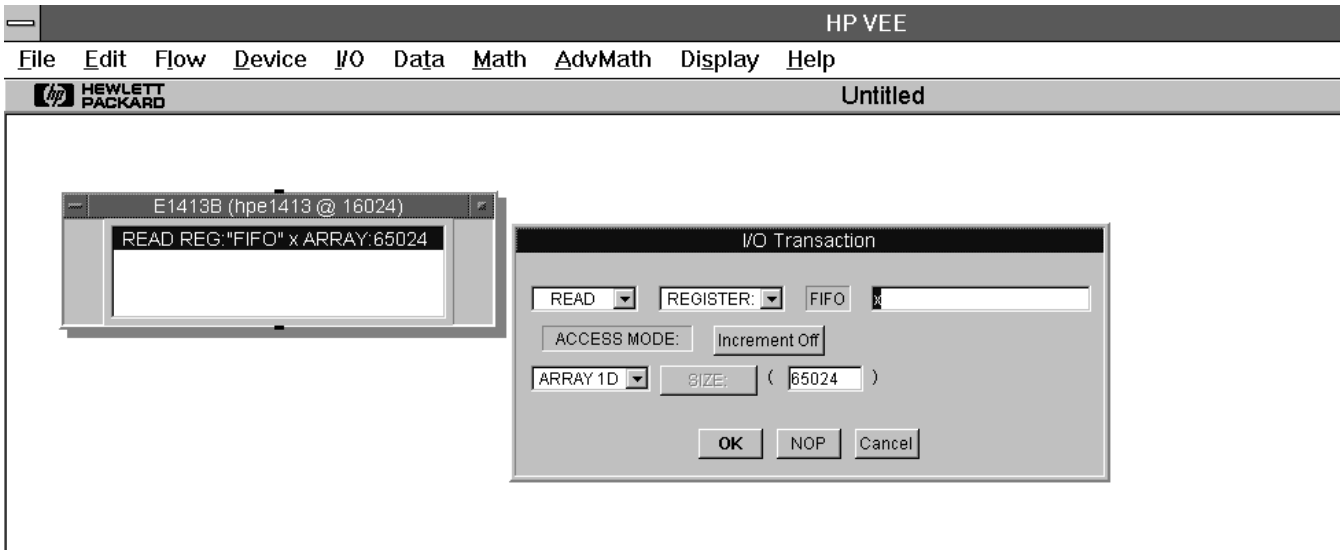
The HP E1313/E1413 current value table is located in A24 address space. Each channel is a 32-bit real number with Channel 0 at offset 0, channel 1 is at offset 4, and so on.

If your hardware supports it, access to both the FIFO and the CVT should use D32 Access as it is twice as fast as D16 access. If using a VXLINK (HP E1383A or HP E1483A) as the interface or if using an HP E1413A, then only D16 access is supported. To use only D16 Access set:

Long Word Access (D32) NONE

3. To read data directly from the FIFO hardware in an HP VEE program, select an HP VEE direct I/O panel for the HP E1313/E1413. In this panel select:

   READ REGISTER FIFO X
   ACCESS MODE Increment off
   ARRAY 1D SIZE 65024

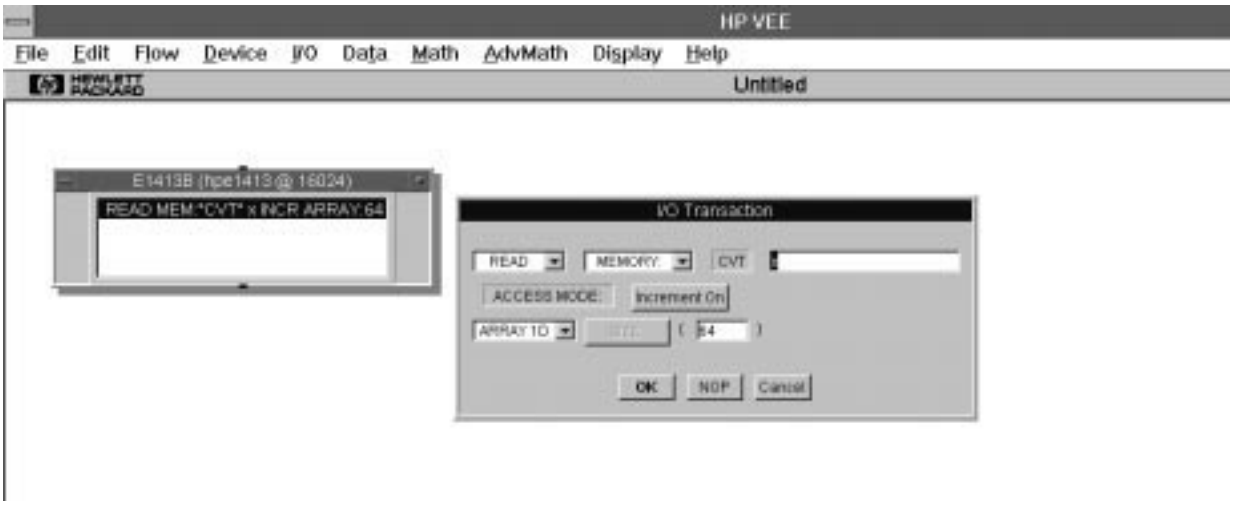


This sets HP VEE up to read the register that in step 2 we named "FIFO" and store the results in variable X.

The Access Mode "Increment off" means that a single location will be read over and over again until the number of readings specified by SIZE is fulfilled. Note that size may be any number up to the maximum size of the HP E1313/E1413 FIFO, which is 65024. An error occurs if you try and enter more readings than are available. Thus, before reading the FIFO using this method, first find out the number of readings that are available.

4. To read data directly from the CVT hardware table in an HP VEE program, select an HP VEE direct I/O panel for the HP E1313/E1413. In this panel select:

   READ MEMORY CVT X
   ACCESS MODE Increment On
   ARRAY 1D SIZE 64

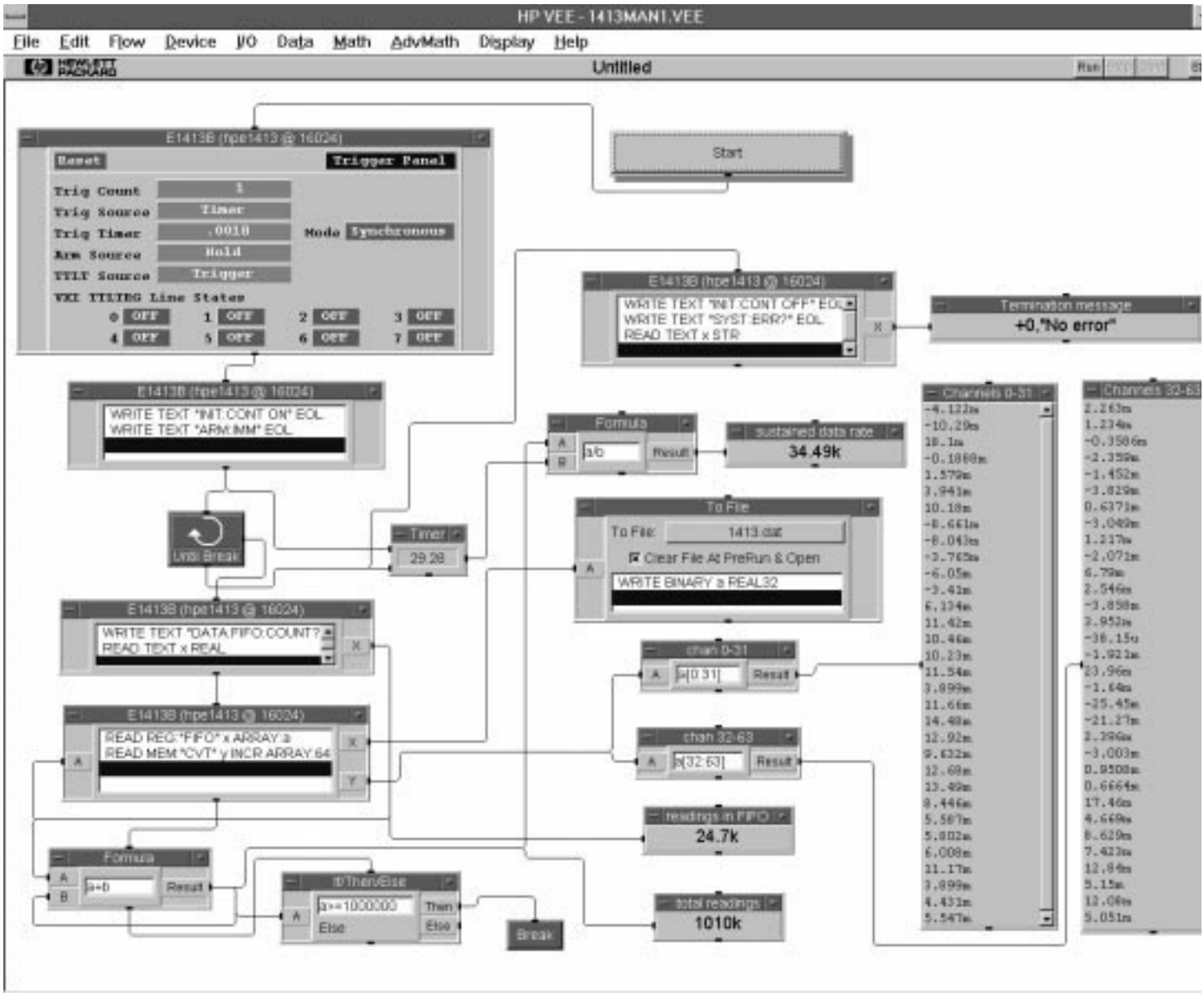


This sets up HP VEE to read the CVT Current Value Memory that in step 2 we named "CVT" and store the results in variable X.

The Access Mode "Increment on" means that successive memory locations will be read to satisfy the number of readings that are specified. In this case, 64 readings have been specified which is the entire CVT table. This results in channel zero as the first element in the 1D array and channel 63 as the 64th element.

Using HP VEE with Direct VXI Backplane Access

1413MAN1.VEE is an example HP VEE program which operates on a computer with direct VXI backplane access. This program configures the HP E1313/E1413 with an instrument panel, then uses Direct I/O SCPI commands to start the collection of data. Direct I/O SCPI commands are used to determine the number of readings in the FIFO and then the FIFO and the CVT are transferred using Direct I/O register access. The instrument panel is shown with it's Trigger sub panel showing. Trig Source is set to Timer which provides good control over the start of each scan, however, this adds overhead and prevents the full 100 K sample rate from being used. Sample Time on the Scan list panel provides control over the time between channels within a scan list. When the instrument panel is executed, the HP E1313/E1413 is configured to match the total state which includes all of the other sub panels that are not visible in this picture.

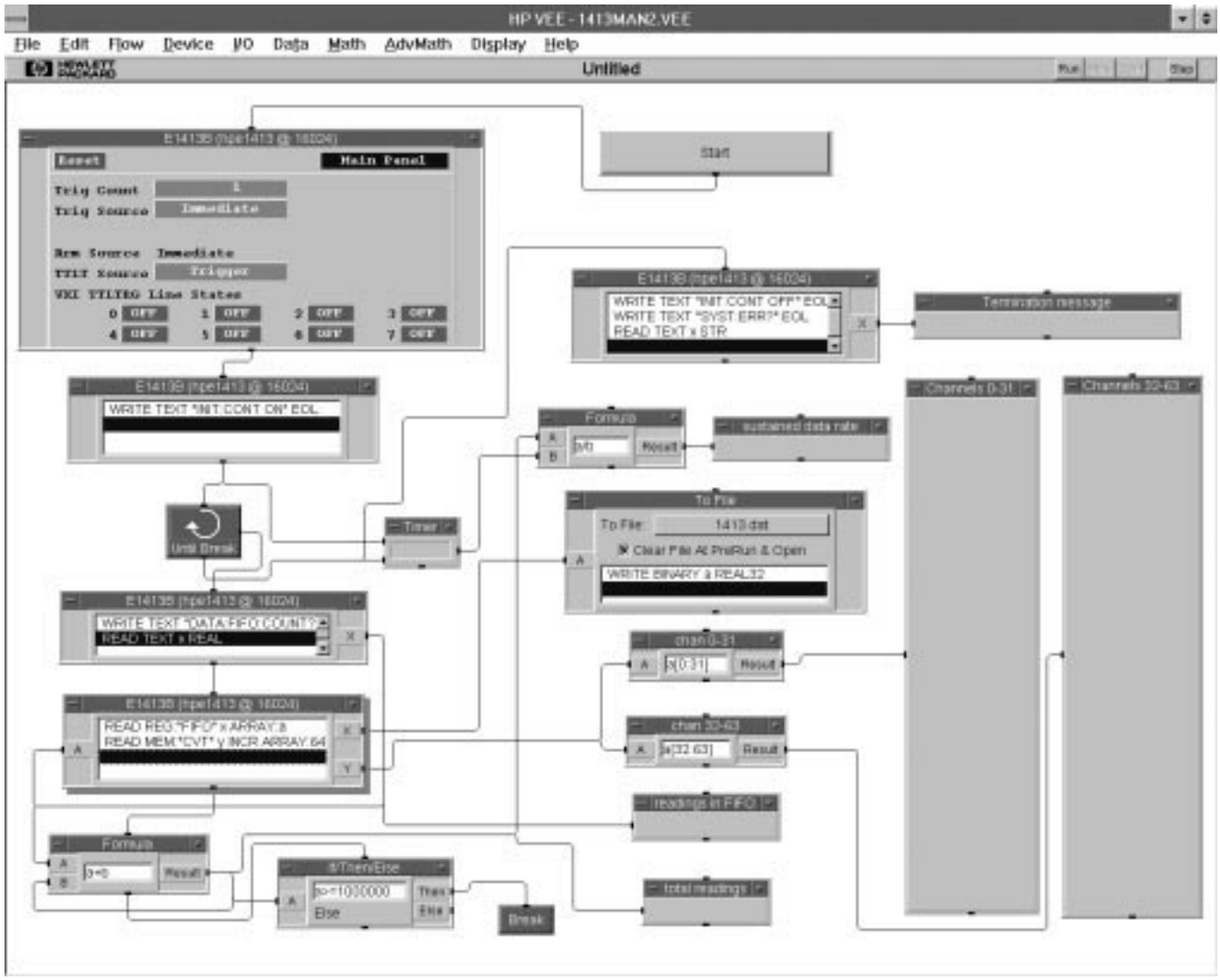


As the program runs, the FIFO is written to a disk file and all of the CVT values are printed each time the FIFO is emptied. The amount of data in the FIFO and the total readings taken are also displayed as a progress monitor. When a programmed break condition occurs, the HP E1313/E1413 is commanded to stop taking readings, and is then queried for any errors that may have occurred. If errors occur, such as "FIFO OVERFLOW", then you need to go back to the instrument panel and adjust parameters such as the Trigger Timer. If "FIFO OVERFLOW" occurs, the timer that calculates a sustained data rate should be used to select a new Trigger Timer value.

A "Trigger too Fast" error occurs if the trigger timer is shortened until there is not enough overhead time to prepare for the next scan. When this happens, switch to the 1413MAN2.VEE program (see next page).

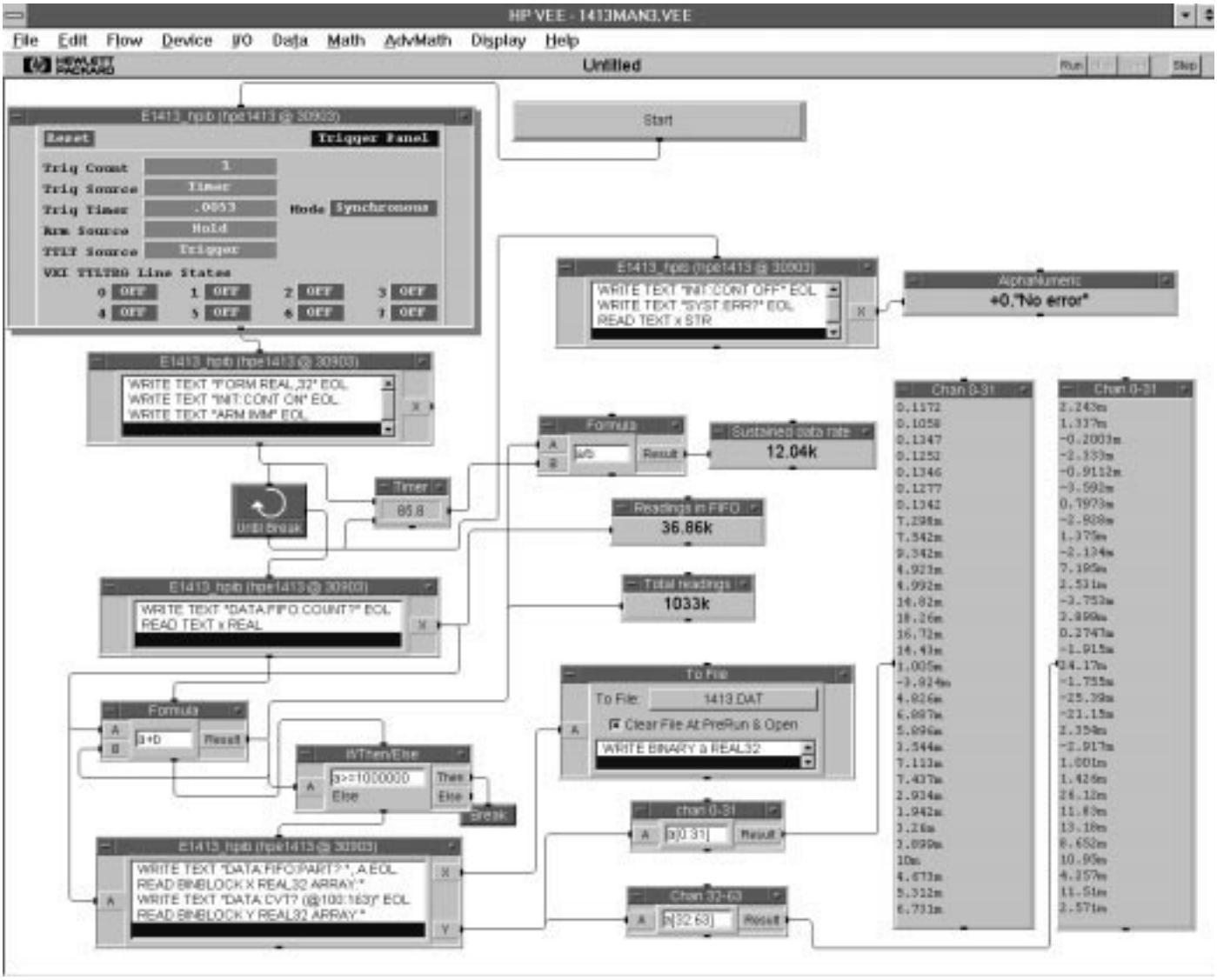
Using HP VEE for 100 K Sample Speed

The program 1413MAN2.VEE provides the full 100 K sample speed. Notice that the Trig Source must be set to Immediate and Arm Source must be Immediate. The Sample Time on the Scan list panel is what controls the pacing of readings.



Using HP VEE with a Command Module

A slower, but still popular system, is one that uses a command module with a HP-IB connection to the computer. In systems that use a command module (i.e. HP E1300/1/6, HP E1405/6), HP VEE's Direct I/O register access cannot be used because there is no direct backplane connection. Instead, the HP E1313/E1413 data must be transferred using Direct I/O SCPI commands as demonstrated in program 1413MAN3.VEE.



Measurement Speeds in HP VEE

The exact maximum sustainable speeds depends a lot on the computer, and the disc hardware used.

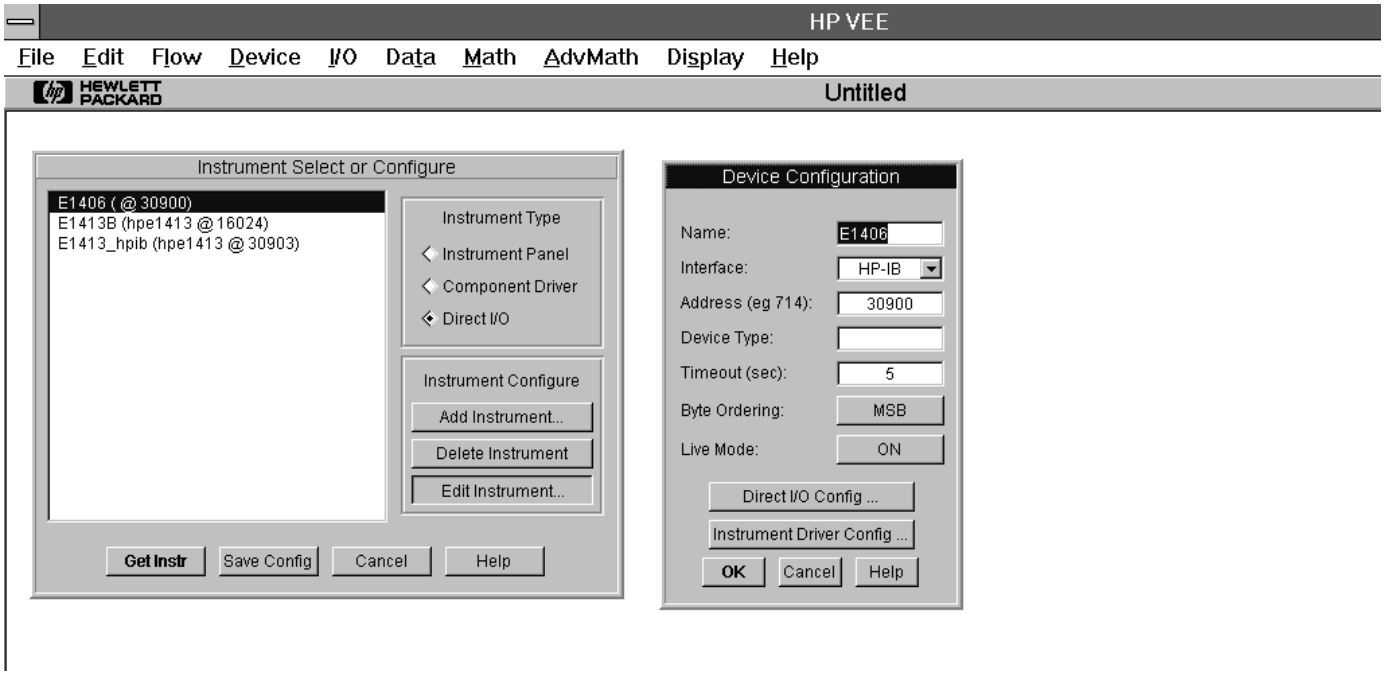
Hardware	Program	Sustained Speed
486/66XM DX2, SCSI disc, hpib, HP E1301	1413MAN4.VEE	9K readings/sec.
486/66XM DX2, SCSI disc, hpib, HP E1406	1413MAN4.VEE	12K readings/sec.
486/66XM DX2, SCSI disc, VXLINK	1413MAN1.VEE	30K readings/sec.
EPC8 486 66Mhz DX2 Embedded	1413MAN1.VEE	50K readings/sec.
EPC7 486 100Mhz DX4 Embedded	1413MAN1.VEE	70K readings/sec.
V743 Embedded	1413MAN2.VEE	100K readings/sec.
745I 100Mhz, MXI	1413MAN2.VEE	100K readings/sec.

None of the above speeds are measured on a real time operating system. Neither Windows or HP-UX is real time, thus, if the operating system or other programs require CPU time, then the above times will not be met.

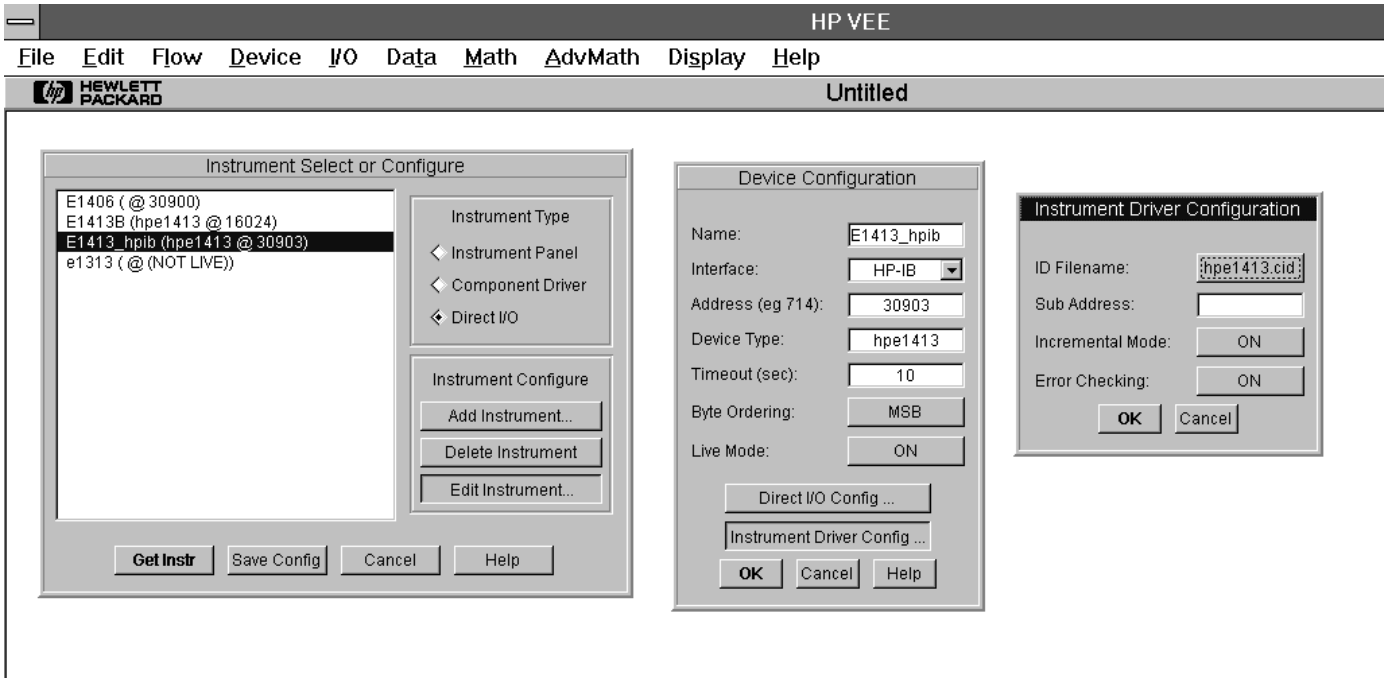
In command module systems, the HP E1313/E1413 SCPI driver has an additional mode of operation that can be used to achieve the full 100K per second speed for short time periods. This mode of operation changes the normal continuous operation to a burst then transfer mode. The command module empties the FIFO and places the readings into VME A24 memory.

The VME memory may be shared A24 memory (<12 Mbyte), or it may be the command module's own memory (<2 Mbyte) that has been configured as NRAM. Each reading stored consumes four bytes. When the command modules memory is to be used, the procedure becomes: configure the command module to have Nram, then use an instrument panel to configure the HP E1413, and finally use Direct I/O to start the measurements and to transfer the results into the computer. This process is demonstrated by the program 1413MAN4.VEE.

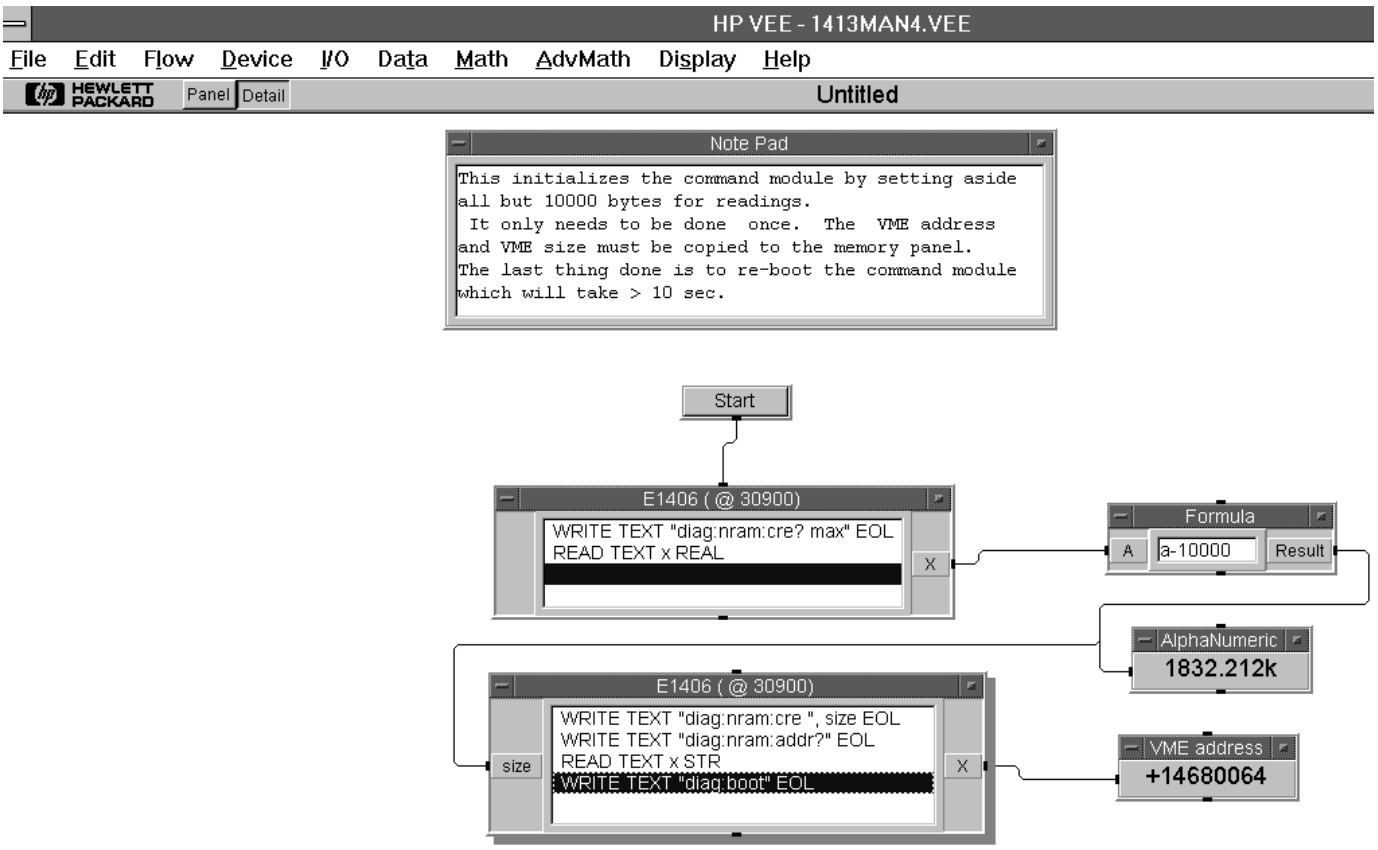
First setup two instruments. The first is a direct I/O to the command module. From the I/O menu:



The second instrument is the HP E1313/E1413. From the I/O menu:

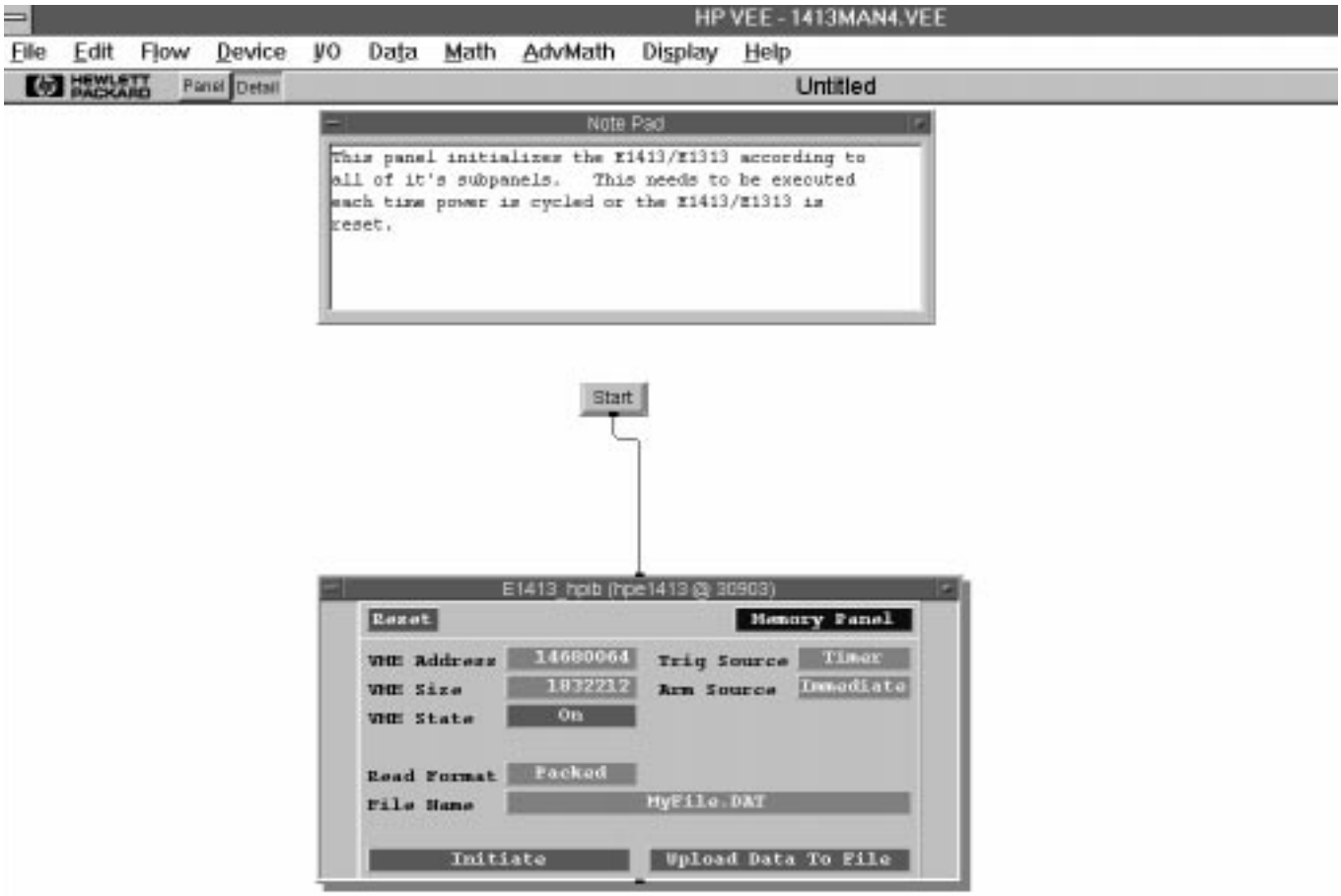


The HP E1406A Command Module should be set so that all of its memory is available. The RAM switches should be set for 2M RAM, 0 Shared. Then the following portion of an HP VEE program configures this memory as Nram. You need to use this procedure because the exact size and address may change based on how an HP E1406A is configured.



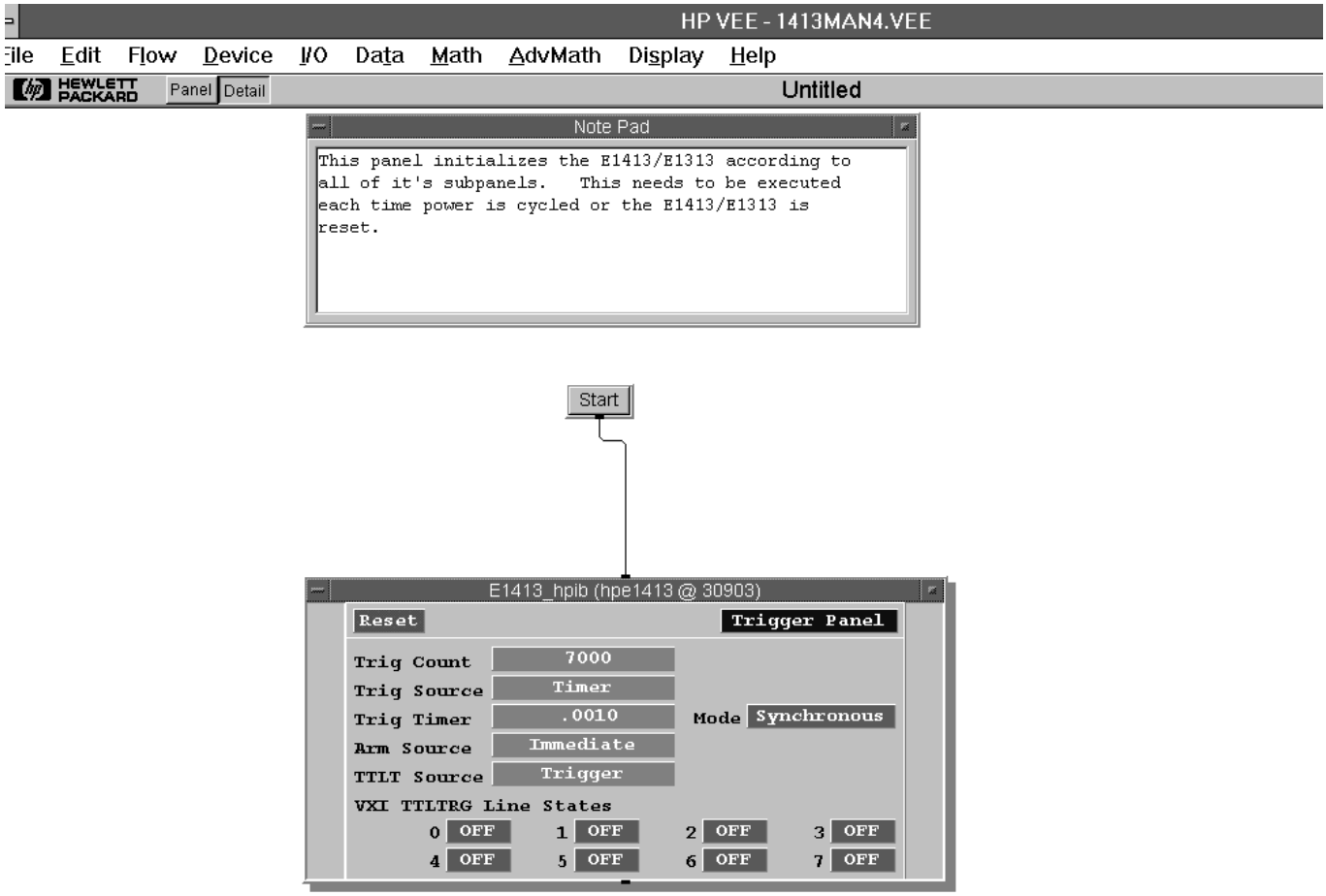
Once the command module has been configured, then use an HP E1413 instrument panel to configure the HP E1313/E1413.

Execute this panel once to configure the HP E1313/E1413. Here is the panel with the memory subpanel showing. Note that the VME Address and VME size has been copied to this panel.



Before executing the above panel, configure various subpanels.

Here is the Trigger subpanel. Notice that 7000 scans will be done at .001 sec intervals, thus this should go for 7 seconds. The Scan list subpanel is set for all 64 channels. Thus, 64 * 7000 = 448,000 readings will be taken which occupy 1,792,000 bytes of memory.

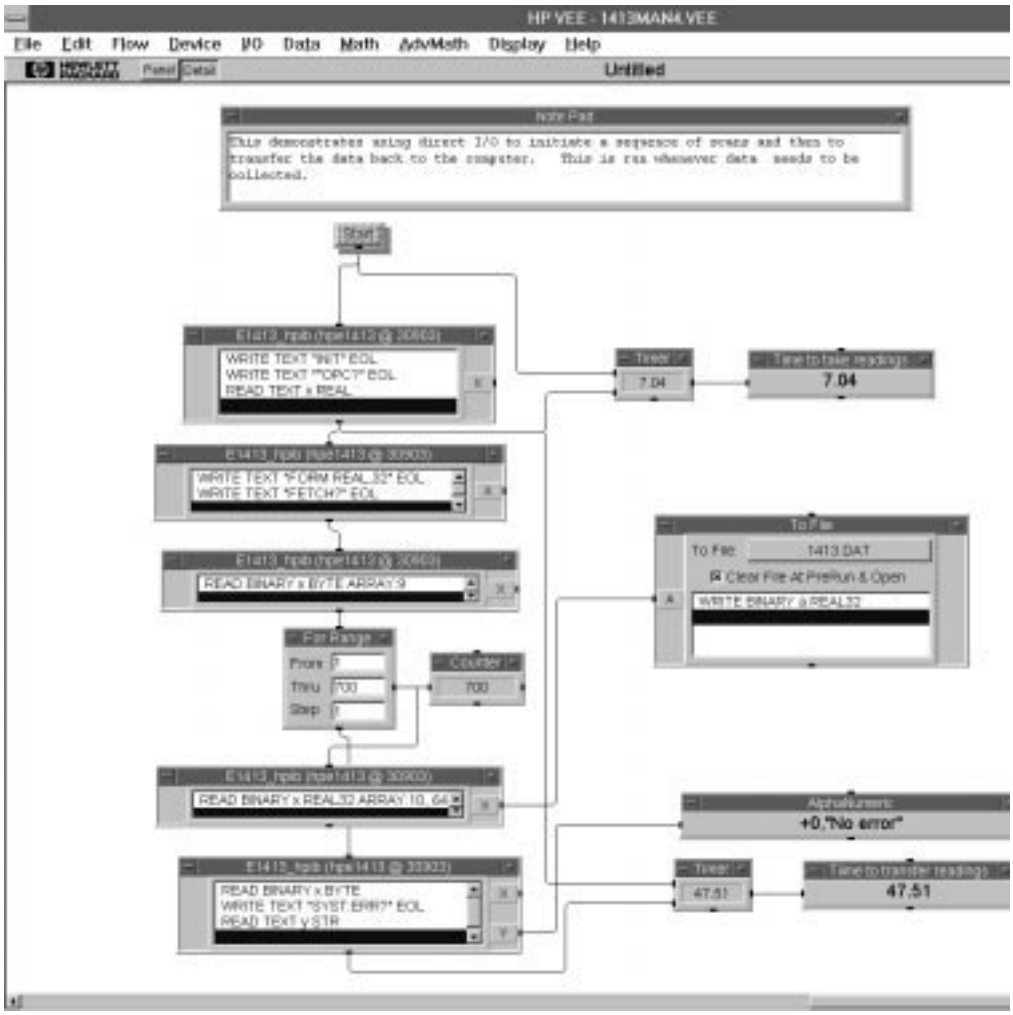


This part of the program starts the measurement and then transfers the readings to the computer. The READ BINARY x BYTE ARRAY:9 reads the IEEE 488.2 arbitrary block header. The block header has the form of #ndd where:

- means block specifier
n - is the number of d to follow
dd - is number of bytes to follow

In this example the header is #71792000, so 9 bytes are read and discarded. Then the readings are read in groups of 10 scans of 64 channels and written to disk. One large READ BINBLOCK command could have been used to read the data, but this would have required a huge amount of free memory in the computer.

Notice that the time to take the readings is much less than the time to transfer the readings into the computer.



Appendix F
Wiring and Noise Reduction Methods

Recommended Wiring and Noise Reduction Techniques

Unshielded signal wiring is very common in Data Acquisition applications. While this worked well for low speed integrating A/D measurements and/or for measuring high level signals, it does not work for high speed sampling A/Ds, particularly when measuring low level signals like thermocouples or strain gage bridge outputs. Unshielded wiring will pick up environmental noise, causing measurement errors. Shielded, twisted-pair signal wiring, although it is expensive, is required for these measurements unless an even more expensive "amplifier at the signal source" or individual A/D at the source is used.

Generally, the shield should be connected to ground at the DUT and left open at the HP E1313/E1413. Floating DUTs or transducers are an exception. Connect the shield to HP E1313/E1413 GND or GRD terminals for this case, whichever gives the best performance. This will usually be the GND terminal. A single point shield to ground connection is required to prevent ground loops. This point should be as near to the noise source as possible and this is usually at the DUT.

Wiring Checklist

The following lists some recommended wiring techniques.

1. Use individually shielded, twisted-pair wiring for each channel.
2. Connect the shield of each wiring pair to the corresponding Guard (G) terminal on the terminal module.
3. The terminal module is shipped with the Ground-Guard (GND-GRD) shorting jumper installed for each channel. These may be left installed or removed, dependent on the following conditions:
    a. **Grounded Transducer with shield connected to ground at the transducer:** Low frequency ground loops (DC and/or 50/60 Hz) can result if the shield is also grounded at the terminal module end. To prevent this, remove the GND-GRD jumper for that channel.
    b. **Floating Transducer with shield connected to the transducer at the source:** In this case, the best performance will most likely be achieved by leaving the GND-GRD jumper in place.
4. In general, the GND-GRD jumper can be left in place unless it is necessary to break low frequency (below 1 kHz) ground loops.

HP E1313/E1413 Guard Connections

The HP E1313/E1413 guard connection provides a 10 KW current limiting resistor between the guard terminals (G) and HP E1313/E1413 chassis ground for each 8 channel SCP bank. This is a safety device for the case where the Device Under Test (DUT) is not actually floating, the shield is connected to the DUT and also connected to the HP E1313/E1413 guard terminal (G). The 10 KΩ resistor limits the ground loop current, which has been known to burn out shields. This also provides 20 KΩ isolation between shields and between SCP banks which helps isolate the noise source.

Common Mode Voltage Limits

You must be very careful not to exceed the maximum common mode voltage referenced to the card chassis ground of æ16 volts (æ60 volts with the HP E1513A Attenuator SCP). There is an exception to this when high frequency (1 kHz - 20 kHz) common mode noise is present (see "HP E1313/E1413 Noise Rejection" later in this appendix). Also, if the DUT is not grounded, then the shield should be connected to the E1313/E1413 chassis ground.

When to Make Shield Connections

It is not always possible to state positively the best shield connection for all cases. Shield performance depends on the noise coupling mechanism which is very difficult to determine. The above recommendations are usually the best wiring method, but if feasible, experiment with shield connections to determine which provides the best performance for your installation and environment.

Note For a thorough, rigorous discussion of measurement noise, shielding, and filtering, see "Noise Reduction Techniques in Electronic Systems" by Henry W. Ott of Bell Laboratories, published by Wiley & Sons, ISBN 0-471-85068-3.

Noise Due to Inadequate Card Grounding

If either or both of the HP E1413 and HP E1482 (MXI Extender Modules) are not securely screwed into the VXIbus mainframe, noise can be generated. Make sure that both screws (top and bottom) are screwed in tight. If not, it is possible that CVT data could be more noisy than FIFO data because the CVT is located in A24 space, the FIFO in A16 space; more lines moving could cause noisier readings.

HP E1313/E1413 Noise Rejection

See Figure F-1 for the following discussion.

Normal Mode Noise (Enm)

This noise is actually present at the signal source and is a differential noise (Hi to Lo). It is what is filtered out by the buffered filters on the HP E1502, HP E1503, HP E1508, and HP E1509 SCPs.

Common Mode Noise (Ecm)

This noise is common to both the Hi and Lo differential signal inputs. Low frequency Ecm is very effectively rejected by a good differential instrumentation amplifier, and it can be averaged out when measured through the Direct Input SCP (HP E1501). However, high frequency Ecm is rectified and generates an offset with the amplifier and filter SCPs (such as HP E1502, HP E1503, HP E1508, and HP E1509). This is since these SCPs have buffer-amplifiers on board and is a characteristic of amplifiers. The best way to deal with this is to prevent the noise from getting into the amplifier.

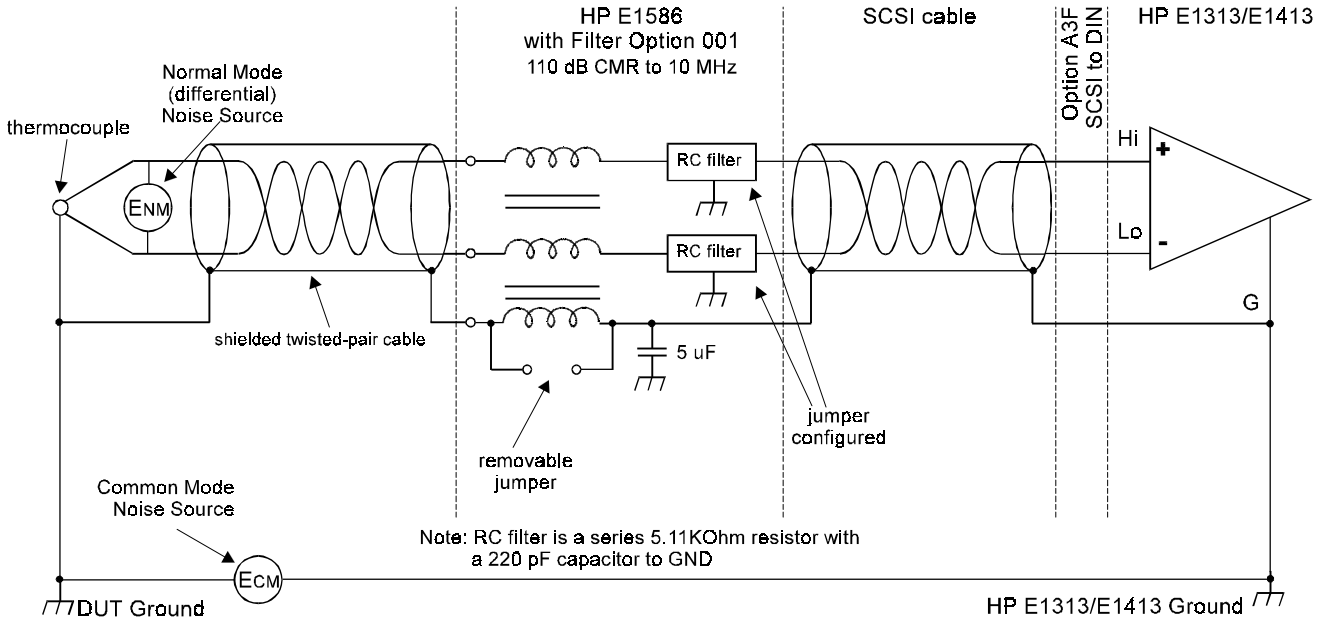


Figure F-1. HF Common Mode Filters

Keeping Common Mode Noise Out of the Amplifier

Most common mode noise is about 60 Hz, so the differential amplifier rejection is very good. The amplifier Common Mode Noise characteristics are:

120 dB flat to 300 Hz, then 20 dB/octave rolloff

The HP E1313/E1413 amplifiers are selected for low gain error, offset, temperature drift, and low power. These characteristics are generally incompatible with good high frequency CMR performance. More expensive, high performance amplifiers can solve this problem, but since they are not required for many systems, HP elected to handle this with the High Frequency Common Mode Filter option to the HP E1586A Rack Mount Terminal Panel (HP E1586 Option 001, HF Filter).

Shielded, twisted-pair lead wire generally does a good job of keeping high frequency common mode noise out of the amplifier, provided the shield is connected to the HP E1313/E1413 chassis ground through a very low impedance. (Not via the guard terminal - The HP E1313/E1413 guard terminal connection shown earlier in this manual does not consider the high frequency Ecm problem, and is there to limit the shield current and to allow the DUT to float up to some DC common mode voltage subject to the maximum ±16 volt input specification limit.)

This conflicts with the often recommended good practice of grounding the shield at the signal source and only at that point to eliminate line frequency ground loops, which can be high enough to burn up a shield. We recommend that you follow this practice, and if you see high frequency common mode noise (or suspect it), tie the shield to the HP E1313/E1413 ground through a 0.1 μF capacitor. At high frequencies, this drives the shield voltage to 0 volts at the HP E1313/E1413 input. Due to inductive coupling to the signal leads, the Ecm voltage on the signal leads is also driven to zero.

Reducing Common Mode Rejection Using Tri-Filar Transformers

One HP E1413 customer determined that greater than 100 dB CMR to 10 MHz was required to get good thermocouple (TC) measurements in his test environment. To accomplish this requires the use of tri-filar transformers which are an option to the HP E1586A Rack Mount Terminal Panel. (This also provides superior isothermal reference block performance for thermocouple measurements.) This works by virtue of the inductance in the shield connected winding presenting a significant impedance to high frequency common mode noise and forcing all the noise voltage to be dropped across the winding. The common mode noise at the input amplifier side of the winding is forced to 0 volts by virtue of the low impedance connection to the HP E1413 ground via the selectable short or parallel combination of 1 kΩ and 0.1 μF. The short cannot be used in situations where there is a very high common mode voltage, (DC and/or AC) that could generate very large shield currents.

The tight coupling through the transformer windings into the signal Hi and Lo leads, forces the common mode noise at the input amplifier side of those windings to 0 volts. This achieves the 110 dB to 10 MHz desired, keeping the high frequency common mode noise out of the amplifier, thus preventing the amplifier from rectifying this into an offset error.

This effectively does the same thing the shielded, twisted-pair cable does, only better. It is especially effective if the shield connection to the HP E1413 ground cannot be a very low impedance due to large DC and/or low frequency common mode voltages.

The tri-filar transformers do not limit the differential (normal mode) signal bandwidth. Thus, removing the requirement for "slowly varying signal

voltages". The nature of the tri-filar transformer, or, more accurately, common-mode inductor, is that it provides a fairly high impedance to common mode signals, and a quite low impedance to differential mode signals. The ratio of common-mode impedance to differential-mode impedance for the transformer we use is ~ 3500:1. Thus, there is NO differential mode bandwidth penalty incurred by using the tri-filar transformers.

Notes

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Notes