

HP 4062UX Semiconductor Process Control System (For S700)

Programming Reference for C Library



HP Part No. 04062-90365
Printed in Japan December 1994

Edition 1
E1294

Product Warranty

This Hewlett-Packard system product is warranted against defects in material and workmanship for a period of one year from date of installation. During the warranty period, HP will, at its option, either repair or replace products which prove to be defective.

Warranty service of this product will be performed at Buyer's facility at no charge within HP service travel areas. Outside HP service travel areas, warranty service will be performed at Buyer's facility only upon HP's prior agreement and Buyer shall pay HP's round trip travel expenses. In all other cases, products must be returned to a service facility designated by HP.

For products returned to HP for warranty service, 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 an instrument will execute its programming instructions when properly installed on that instrument. HP does not warrant that the operation of the instrument, or software, or firmware will be uninterrupted or error free.

Limitation of Warranty

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

NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HEWLETT-PACKARD SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Exclusive Remedies

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

Assistance

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

Certification

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

Copyright © 1994 Hewlett-Packard Company.

Printing History

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by the customer. The dates on the title page change only when a new edition is published. No information is incorporated into a reprinting unless it appears as a prior update; the edition does not change when an update is incorporated.

The software revision code printed before the date indicates the version level of the software product at the time the manual or update was issued. Many product updates and fixes do not require manual changes and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one to one correspondence between product updates and manual updates.

Edition 1

December 1994 for Revision X.07.0*x*

Safety Summary

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. 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 instrument. Hewlett-Packard Company assumes no liability for customer's failure to comply with these requirements.

GROUND THE INSTRUMENT

To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The power terminal and the power cable must meet International Electrotechnical Commission (IEC) safety standards.

DO NOT OPERATE IN AN EXPLOSIVE ATMOSPHERE

Do not operate the instrument in the presence of flammable gases or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

KEEP AWAY FROM LIVE CIRCUITS

Operation personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

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 INSTRUMENT

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

DANGEROUS PROCEDURE WARNINGS

Warnings, such as example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed.

Warning



DANGEROUS VOLTAGES, CAPABLE OF CAUSING DEATH, ARE PRESENT IN THIS INSTRUMENT. USE EXTREME CAUTION WHEN HANDLING, TESTING, AND ADJUSTING.

Safety Symbols

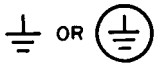
The general definitions of safety symbols used on equipment or in manuals are listed below.



Instruction manual symbol: the product is marked with this symbol when it is necessary for the user to refer to the instruction manual in order to protect against damage to the instrument.



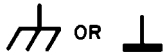
Indicates dangerous voltage (terminals fed from the interior by voltage exceeding 1000 volts must be so marked).



Protective conductor terminal. For protection against electrical shock in case of a fault. Used with field wiring terminals to indicate the terminal which must be connected to ground before operating equipment.



Low-noise or noiseless, clean ground (earth) terminal. Used for a signal common, as well as providing protection against electrical shock in case of a fault. A terminal marked with this symbol must be connected to ground in the manner described in the installation (operation) manual, and before operating the equipment.



Frame or chassis terminal. A connection to the frame (chassis) of the equipment which normally includes all exposed metal structures.



Alternating current (power line).



Direct current (power line).



Alternating or direct current (power line).

Warning



The warning sign denotes a hazard. It calls attention to a procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in injury or death to personnel.

Caution



The caution sign denotes a hazard. It calls attention to an operating procedure, practice, condition or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product.

Note



The note sign denotes important information. It calls attention to a procedure, practice, condition or the like, which is essential to highlight.

Preface

Purpose

This manual contains the information about functions that are provided for the C programming environment for the HP 4062UX system installed in series 700 computers.

Audience

This manual is intended for programmers of the HP 4062UX system, who are familiar with C programming in an HP-UX environment.

Contents

This manual provides the information on all the functions that is used in the user-written test programs.

Contents

1. General Information

Introduction	1-1
How to Use the Reference Pages	1-1
Function Name	1-2
Synopsis	1-2
Arguments	1-2
Description	1-2
Example	1-3
See Also	1-3
Assumptions on the Reference Pages	1-3
Abbreviations	1-3
Software Support	1-4
Pcsstart	1-5

2. HP-IB Control

Introduction	2-1
Open_tis_hpib_fd	2-2
Set_tis_hpib_fd	2-3

3. System Initialization and Locking

Introduction	3-1
System Initialization	3-1
System Locking	3-1
Init_system	3-2
Lock_system	3-4
Unlock_system	3-6

4. Switching Matrix Control

Introduction	4-1
Connect_pin	4-2
Connect_th	4-5
Disconnect_all	4-7

5. General I-V Measurements

Introduction	5-1
Disable_port	5-2
Disable_port_all	5-4
Force_i	5-5
Force_meas	5-8
Force_v	5-12
Long_cal	5-18
Measure_i	5-19

Measure_p	5-22
Measure_v	5-26
Port_status	5-30
Set_pbias	5-34
Set_smu	5-41
Set_vm	5-43
Short_cal	5-45

6. Sweep Measurements

Introduction	6-1
Rsweep_iv	6-2
Rsweep_miv	6-9
Rsweep_stop	6-14
Set_iv	6-15
Set_piv	6-23
Set_sync (for sweep measurements)	6-30
Sweep_iv	6-36
Sweep_miv	6-47

7. Search Measurements

Introduction	7-1
Search_iv	7-2
Set_asearch	7-5
Set_bsearch	7-12
Set_lsearch	7-16
Set_sync (for search measurements)	7-20

8. Quasi-Pulsed Measurements

Introduction	8-1
Measure_bdv	8-2
Measure_ileak	8-4
Set_bdv	8-7
Set_ileak	8-10

9. DC Source Output Relay Control

Introduction	9-1
Ch_sw_off	9-2
Ch_sw_off_all	9-4
Ch_sw_on	9-6
Ch_sw_on_all	9-8

10. Capacitance Measurements

Introduction	10-1
Measure_cmu	10-2
Measure_cmu84	10-4
Open_corr84	10-7
Set_cmu	10-8
Set_cmu84	10-10
Set_ct	10-12
Set_cv	10-15
Set_cv84	10-18

Set_freq	10-20
Short_corr84	10-22
Sweep_ct	10-23
Sweep_cv	10-27
Sweep_cv84	10-30
11. Offline Debugging and Run-time Error Handling	
Introduction	11-1
Offline Debugging	11-1
Run-time Error Handling	11-1
Data_creation	11-2
Error_info	11-3
Is_on_line	11-4
Off_line_data	11-5
Tis_errno	11-6
12. Port and Unit Addresses	
Introduction	12-1
Fnaux	12-2
Fncmh	12-4
Fncml	12-5
Fncmu	12-6
Fncmu84	12-7
Fngcmu	12-8
Fngnd	12-9
Fngsmu	12-10
Fnport	12-11
Fnsmu	12-13
Fnunitsmu	12-14
Fnvm	12-15
Fnvs	12-17
13. C Library Prober Drivers	
Introduction	13-1
P_down	13-3
P_home	13-4
P_imove	13-7
P_ink	13-9
P_move	13-12
P_orig	13-13
P_pos	13-15
P_scale	13-16
P_up	13-17
Prober_enter	13-18
Prober_get_ba	13-20
Prober_get_eid	13-21
Prober_get_name	13-22
Prober_identify	13-23
Prober_init	13-24
Prober_open	13-26
Prober_output	13-27

Prober_read_sysconfig	13-29
Prober_reset	13-30
Prober_spoll	13-31
Prober_status	13-32
Prober_wait	13-34
Prober_waittime_ctl	13-35

14. Probing Control and File Creation Libraries

Introduction	14-1
The Probing Control Library (PCL)	14-1
The File Creation Library (FCL)	14-1
W_build_file	14-3
W_file_type	14-5
W_get	14-6
W_move_next	14-7
W_move_number	14-8
W_move_xy	14-9
W_pos_number	14-10
W_pos_xy	14-11
W_put_array	14-12
W_put_data	14-14
W_read_file	14-16
W_return_limit	14-18
W_return_number	14-19
W_return_xy	14-20
W_save_data	14-21
W_start_number	14-22
W_start_xy	14-24
W_total_chip	14-26

15. DCS Program Memory Library

Introduction	15-1
Fnaddr4142	15-2
Fnchannel4142	15-3
Fnrange4142	15-4
Fnunit4142	15-6
Pcs_hpib_clear	15-7
Pcs_hpib_enter	15-8
Pcs_hpib_open	15-10
Pcs_hpib_output	15-11
Pcs_hpib_spoll	15-13
Pgm_ch_sw_off	15-14
Pgm_ch_sw_on	15-16
Pgm_disable_dcs	15-18
Pgm_end	15-20
Pgm_force_i	15-21
Pgm_force_v	15-23
Pgm_measure_i	15-25
Pgm_measure_p	15-27
Pgm_measure_v	15-29
Pgm_search_iv	15-31

Pgm_set_asearch	15-32
Pgm_set_pbias	15-36
Pgm_set_smu	15-39
Pgm_set_vm	15-40
Pgm_start	15-41
Pgm_wait	15-42
Readerror4142	15-43
Readout_dcs	15-45
Readsweep_dcs	15-47
Run_dcs_program	15-50

A. Error Messages

Introduction	A-1
Pcsstart Errors	A-1
TIS Errors	A-10
PCL/FCL Errors	A-17
PGMLIB Errors	A-18
Prober Driver Library Errors	A-19

Index

Tables

1-1. 4062 TISC Fileset Contents 1-4

General Information

Introduction

This chapter provides general information needed to use the C Library and this manual. This chapter includes the following information:

- Introduction
- How to Use the Reference Pages
- Assumptions on the Reference Pages
- Software Support
- Pcsstart

The C Library is an optional product of the HP 4062UX. This option includes the C Library Test Instruction Set (TIS), the Prober Drivers, and the Probing Control Library (PCL) and File Control Library (FCL).

This manual describes the functions available in the C Library and the C wafer prober libraries of the HP 4062UX. This manual describes all the TIS, WAFER, and prober driver functions included in your HP 4062UX system software.

How to Use the Reference Pages

This manual is designed so you can look up information about the C Library by function name. It defines the parts of the functions and explains how they work.

The functions are categorized according to the tasks they perform. The tasks in which the functions are arranged follow:

- HP-IB Control
- System Initialization and Locking
- Switching Matrix Control
- General I-V Measurements
- Sweep Measurements
- Search Measurements
- DC Source Output Relay Control
- Capacitance Measurements
- Offline Debugging and Run-time Error Handling
- Port and Unit Addresses
- C Library Prober Drivers
- Probing Control and File Creation Libraries

The functions within the sections above are arranged in alphabetical order. Each function is arranged into a description module. The description modules are arranged in the following format:

- Function name
- Synopsis
- Arguments
- Description
- Example
- See Also

Not all of the above topics apply to every function and therefore are not included in some description modules.

Function Name

The function name is listed at the top of the description module with a brief description. Each function is listed in alphabetical order within the chapter in which it is located.

Synopsis

The synopsis of the function is indicated by **computer** type. The function is a primary expression followed by parenthesis containing possibly empty, comma-separated list of expressions that constitute the actual arguments to the function.

The following is an example of a synopsis entry:

```
int force_v(port, voltage, range, compliance)
int port;
double voltage, range, compliance;
```

Arguments

The argument of a function specifies the computing operations that are to be done. The arguments are listed and described briefly in a table. For more extensive information about the arguments, see the Description section for that function. The arguments are indicated by *italic* type in the text.

The C Library provides several macros that can be used with the function arguments. The macros are defined in this section as well. By convention, the macros are indicated by UPPER CASE type. A macro must be specified by UPPER CASE letters whenever it is used in a program.

Description

The description provides detailed information about the function, including information about offline mode, compliance values, and return values. Along with text, the description may include tables and figures.

Example

The Example section provides an example of the function. The example is taken from an actual program that uses the function. The example is indicated by **computer** type.

See Also

The See Also section provides a list of related C Library functions.

Assumptions on the Reference Pages

To simplify the function descriptions, certain information is assumed. This section lists the information that you should be familiar with to effectively use this manual.

This manual assumes that you are familiar with the HP 4062UX system. Please see the HP 4062UX system manuals if you are not familiar with the system.

This manual also assumes that you are familiar with the C programming language.

Abbreviations

Throughout this manual, the following abbreviations are used to identify subsystems and instruments of the HP 4062C/UX:

SMS:	Switching Matrix Subsystem: consist of the SWC(s) and SWM
SWC:	HP 4084B Switching Matrix Controller
SWM:	HP 4085A/B or HP 4089A/B Switching Matrix
DCS:	DC Measurement Subsystem: HP 4142B Modular DC Source/Monitor
CMS:	Capacitance Measurement Subsystem
CMU:	Capacitance Measurement Unit: HP 4280A 1 MHz C Meter/C-V Plotter
CMU84:	Capacitance Measurement Unit: HP 4284A 20 Hz–1 MHz Precision LCR Meter
SMU:	Source Monitor Unit: HP 41420A (1 A/200 V) and HP 41421B (100 mA/100 V)
VS/VMU:	Voltage Source/Voltage Monitor Unit: HP 41424A
AFU:	Analog Feedback Unit: HP 41425A
GNDU:	Ground Unit
TIS:	Test Instruction Set

Software Support

This section provides tables that show software supported by the C Library. It also includes information about Pcsstart, which is an HP 4062UX command used to initialize the system.

Table 1-1. 4062 TISC Fileset Contents

File Name	Description
/usr/pcs/lib/libhp4062.a	The C Library file. Also consists of PCL/FCL and some wafer prober support procedures.
/usr/pcs/lib/libprober.a	The part of the prober driver library consisting of the shared code among all HP supported prober models. This file must be linked whenever you use the HP supported prober driver libraries.
/usr/pcs/lib/libegx.a /usr/pcs/lib/libegy.a /usr/pcs/lib/libkla.a /usr/pcs/lib/libtel.a /usr/pcs/lib/libtsk.a /usr/pcs/lib/libmjc.a	The part of the prober driver library consisting of code for specific prober models. Note that more than one library can be linked to one object file at a time. This lets the object file operate on any prober model in which the driver is linked.
/usr/include/pcs/tis.h	This include file consists of the C Library function declarations and the macro definitions.
/usr/include/pcs/prober.h	This include file consists of the C wafer prober driver functions and variable declarations.
/usr/include/pcs/wafer.h	This include file consists of the PCL/FCL functions and variable declarations. This file also provides the C language structure tag definitions for PPF or FCL created data file access.
/usr/pcs/demo/c/spot.c /usr/pcs/demo/c/sweep.c /usr/pcs/demo/c/search.c /usr/pcs/demo/c/cg.c /usr/pcs/demo/c/cg84.c	These are sample C programs that illustrate the following measurements: dc spot I/V , dc I/V sweep measurement, transistor hfe extraction using the analog search method, and capacitance/conductance.
/usr/pcs/src/prober	A directory that contains the prober driver library source files.
/usr/pcs/src/wafer	A directory that contains the PCL/FCL library source files.

Pcsstart

This is an HP 4062UX command that initializes the semiconductor parametric test session and re-starts it with a given execution mode (online or offline).

Synopsis

```
pcsstart [-q] [-v] { -off [file] } [-sysconfig file]
```

Options

- | | |
|-------------------------------|---|
| -q | Used for quick operation, this option eliminates DCS calibration. The option saves command execution time. Although pcsstart does not do calibrations, system components are all left in the initialized state after this command. |
| -v | Used for verbose mode. Prints system configuration to standard output. |
| -off [<i>file</i>] | Switches the parametric test session to offline mode. Test execution is in the offline mode until the pcsstart command is used to reset the mode to online by the -on option. An optional <i>file</i> gives a name of a file containing system configuration information. If a <i>file</i> is not specified, the <code>/usr/pcs/etc/DUMCONFIG</code> file is used. This option can not be used with the -on option. |
| -on | Switches the parametric test session to online mode. Test execution is in online mode until pcsstart is used to reset the mode to offline by the -off option. This option cannot be used with the -off option. |
| -sysconfig <i>file</i> | Specifies the alternative SYSCONFIG file that overrides the default <code>/usr/pcs/etc/SYSCONFIG</code> file. |

Pcsstart

Description

Pcsstart initializes a semiconductor parametric test session and re-starts it with either the online or the offline execution mode. The command reads the value of the shell environment variable *PCS_SESSION_ID* as a session name to be re-started. If the session is in the online mode, or if the pcsstart resets the execution mode to online mode, test system hardware initialization and configuration checking is performed.

If neither the **-on** or **-off** option is specified, the session does not change the execution mode. The default execution mode is set to offline.

Pcsstart does not initialize wafer probers.

During pcsstart execution, the test system instruments are locked. If some other process is locking the instruments, Pcsstart fails and reports an error.

HP-IB Control

Introduction

The C Library provides the following functions for HP-IB control:

- `Open_tis_hpib_fd`
- `Set_tis_hpib_fd`

All the system components in the HP 4062UX system are controlled through an HP-IB. HP-IB is accessed by using Standard Instrument Control Library (SICL). To use the SICL library, you must use `#include` at the beginning of your test programs to include the `/usr/lib/sicl.h` file.

Open_tis_hpib_fd

Scans the `/usr/pcs/etc/SYSCONFIG` file to find the HP-IB logical unit number for the system instruments.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

INST open_tis_hpib_fd()
```

Description

This function scans the `/usr/pcs/etc/SYSCONFIG` file to find the logical unit number. When the logical unit number is found, the function returns a session ID.

This procedure returns a `-1` if an error is detected.

Example

```
. . .
INST fd;
. . .
if ((fd = open_tis_hpib_fd()) == -1) err_rep();
set_tis_hpib_fd(fd);
. . .
. . .
```

See Also

Set_tis_hpib_fd

Set_tis_hpib_fd

Allows the C Library TIS to use the session ID *fd* for HP-IB access via the SICL.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

int set_tis_hpib_fd(fd)
INST fd;
```

Arguments

Item	Range Restriction/Description
<i>fd</i>	Specifies a session ID obtained from the Open_tis_hpib_fd function in the C Library, or from the iopen(3) SICL function call.

Description

The session ID *fd* allows the SICL to access any instrument through the HP-IB interface. The *fd* argument must refer to an HP-IB interface card that is defined in the SYSCONFIG file. This function must be called before any C Library operation that uses the HP-IB interface for system instruments.

Both read and write operations are allowed with this session ID. A session ID returned from the Open_tis_hpib_fd function always satisfies the necessary conditions.

If the session ID has been closed, reopen and pass it to the C Library again by this function before using any additional C Library function.

Example

```
. . .
INST fd;
. . .
if ((fd = open_tis_hpib_fd()) == -1) err_rep();
set_tis_hpib_fd(fd);
. . .
. . .
```

See Also

Open_tis_hpib_fd

System Initialization and Locking

Introduction

This chapter describes the System Initialization and System Locking tasks of the C Library. The functions for these tasks are listed alphabetically in this chapter.

System Initialization

The C Library provides the following functions to initialize the system:

- `Init_system`

When you turn the HP 4062UX on, the initial control settings of each system instrument are slightly different from the control settings established during a system initialization. When you execute system software, each instrument must be initialized to satisfy system requirements. Therefore, it is strongly recommended that you use the `Init_system` function at the beginning of all test programs written with the C Library.

System Locking

The C Library provides the following functions to lock the system instruments:

- `Lock_system`
- `Unlock_system`

Whenever the system resources are explicitly locked to a process, no other process can access the system resources. The functions described in this section are provided to explicitly lock and unlock the system resources to a single process. The `Lock_system` function locks the system resources until the `Unlock_system` function is called or until you exit the C Library program.

For the best test results, it is strongly recommended that you use the `Lock_system` function at the beginning of any C Library program.

Note



The C Library `Init_system` does not reset the wafer prober driver library. `Prober_init` must be used before any prober library or PCL functions that access the wafer prober.

Init_system

This function initializes the system (excluding the wafer prober).

Synopsis

```
#include <pcs/tis.h>
```

```
int init_system()
```

Description

Execute Init_system before executing any other TIS function. Init_system clears all previous settings and sets the main functions of the systems (excluding the wafer prober) as follows:

Subsystem	Item	Settings
SWM	Measurement Pins	Disconnected
DCS	SMU	VS Mode, 20 V Range, 0 V Output, 100 μ A Current Compliance, Filter ON
	VM	20 V Range, Grounded Measurement mode
	VS	0 V Output, 20 V Range
	Integration Time	SHORT
	Auto Calibration	ON
CMU	Output Switch	OFF
	C-G Range	Auto Range
	Test Signal Level	30 mVrms
	Integration Time	MEDIUM
CMU84	Bias Source	0 V Output, INTERNAL DC BIAS, V Output Lamp ON (enable)
	Integration Time	MEDIUM
	Measurement Range	AUTO
	Test Signal Level	30 mVrms
	DC Bias Source	0 V Output
	Frequency	1 MHz

Init_system also clears the measurement parameters set by the following functions:

- Set_asearch
- Set_bsearch
- Set_cmu84
- Set_ct
- Set_cv
- Set_cv84
- Set_freq
- Set_iv
- Set_lsearch
- Set_pbias
- Set_piv
- Set_sync

The initial control settings of the SWM, DCS, CMU, and CMU84 can be found in chapter 5 of the *System Information Manual*.

Example

```
. . .  
INST fd;  
. . .  
. . .  
set_tis_hpib_fd(fd);  
if (init_system() == -1) error_rep();  
. . .  
. . .
```

Lock_system

This function explicitly locks the system instruments that are specified in a calling process to that process.

Synopsis

```
#include <pcs/tis.h>

int lock_system(wait)
int wait;
```

Argument

Item	Range Restriction/Description
<i>wait</i>	Determines the behavior of the function when it is not able to lock the system instruments to the calling process. No meaning if system instruments are locked by another session. Specify 0 or 1.

Description

This function automatically locks the system. Examine the return value and the TIS error code to check the status of the session.

Either 0 or 1 can be specified for the *wait* argument:

- 0 Returns immediately regardless of session state. If the session is busy, this procedure returns `-1` to indicate the failure of the locking request, and `Tis_errno` is set to 202 (session locked).
- 1 If the session is locked to another process, the function waits until the session is released. The calling process is set to sleep status while it waits for the session to be free.

In Offline Mode:

Lock_system is ignored in offline mode execution.

Example

```
. . .  
. . .  
if (lock_system(0) == -1) error_rep();  
. . .  
. . .
```

See Also

Unlock_system

Unlock_system

This function unlocks explicitly locked system instruments.

Synopsis

```
#include <pcs/tis.h>

int unlock_system()
```

Description

This function unlocks the system instruments from the calling process. If the system instruments are not explicitly locked when this function executes, this function is simply ignored.

Explicit locking cannot be nested.

In Offline Mode:

Unlock_system is ignored in the offline mode.

Example

```
. . .
if (unlock_system() == -1) error_rep();
. . .
```

See Also

Lock_system

Switching Matrix Control

Introduction

The C Library provides the following functions to connect measurement pins to ports:

- Connect_pin
- Connect_pin2
- Connect_pin3
- Connect_th
- Disconnect_all

Throughout this chapter, these functions are sometimes collectively referred to as the Connect_pin functions.

These functions connect or disconnect a specified measurement port to specified measurement pins. The first three functions implement the connection of one, two, or three pins. To connect more than three pins to one port at one time, repeat any of the connection functions.

To disconnect all the pins and ports, either use the Disconnect_all function or specify 0 for both the *pin* and *port* arguments in the Connect_pin function. Specifying 0 for a *pin* argument disconnects the specified port from any pins. Specifying 0 for a *port* disconnects the specified pins from any ports.

If the same pin number appears in two or more Connect_pin functions that specify different ports, only the port specified in the last Connect_pin function is connected to the measurement pin.

The Connect_pin, Connect_pin2, and Connect_pin3 functions are combined in this chapter and presented under the single heading Connect_pin.

Please see the Port and Unit Addresses chapter of this manual for more details on the addresses of ports.

The functions are listed alphabetically in this chapter.

Connect_pin

These functions connect a specified port to a specified measurement pin.

Synopsis

```
#include <pcs/tis.h>

int connect_pin(port, pin)
int port, pin;

int connect_pin2(port, pin1, pin2)
int port, pin1, pin2;

int connect_pin3(port, pin1, pin2, pin3)
int port, pin1, pin2, pin3;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address to be connected to the specified <i>pin</i> . 0, 32701 to 32711, 32721 to 32732 ¹
<i>pin</i> , <i>pin1</i> , <i>pin2</i> , <i>pin3</i>	Specifies an SWM pin number to be connected to the specified <i>port</i> . 0 through 96 ²

¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 0 through 48 for HP 4085B 48-pin SWM systems.

Description

There are three functions for one, two, or three pin connections: Connect_pin, Connect_pin2, and Connect_pin3. To connect more than three pins to one port at one time, add pin connections to existing ones by repeatedly calling one of the connection functions.

These functions connect a specified port to a specified measurement pin. No more than one port can be connected to a measurement pin. Thus, if the same *pin* appears in two or more connect functions that specify different ports, only the port specified in the last Connect_pin function is connected to the measurement pin.

A value returned from Fnport, Fnsmu, Fnaux, Fncmh, Fncml, Fngnd, Fnsgmu, or Fngcmu is suitable for the *port* argument.

If *port* is 0, the measurement pin specified in the function is disconnected. Also, specifying 0 for the *pin number* disconnects the specified port from all measurement pins. If both *port* and *pin* are 0, all measurement pins are disconnected from all ports.

If the device connected to the specified port is ACTIVE when a Connect_pin function executes, other related devices are set to Zero Output. Refer to the following table:

4-2 Switching Matrix Control

Port	Port Number	Description
SMU1	1	Only the specified SMU is set to Zero Output status if it is ACTIVE when a Connect_pin function executes. The status of other ports, ACTIVE or not, is not affected.
SMU2	2	
SMU3	3	
SMU4	4	
AUX33(SMU) ¹	29	
AUX43(SMU) ¹	32	
AUX1	5	If the VS cable is connected to these ports, all ACTIVE VSs and SMUs connected to the SWM are set to Zero Output.
AUX2	6	
AUX11 ¹	21	
AUX12 ¹	22	
AUX13 ¹	23	
AUX21 ¹	24	
AUX22 ¹	25	
AUX23 ¹	26	
AUX32 ¹	28	
AUX42 ¹	31	
GNDU	7	All ACTIVE SMUs and VSs are set to Zero Output.
GSMU	10, -1	
AUX3(CMH)	8	If the VS cable is connected to these ports, all ACTIVE VSs and SMUs connected to the SWM are set to Zero Output.
AUX4(CML)	9	
GCMU	11, -8, -9	If CMH and CML cables are connected to these ports, the internal dc bias source of the CMU is set to Zero Output.
AUX31(CMH) ¹	27	
AUX41(CML) ¹	30	

¹ For the SWM equipped with a port expander.

Note



If you connect the terminals of unsupported instruments to the AUX ports, use dry-switching. Before executing any Connect_pin function, confirm that unsupported instruments force 0 V.

If a device is set to zero output status by a Connect_pin function and is still connected to measurement pins, it can be returned to its previous setting with the Force_i or Force_v function.

If your system is equipped with an HP 4089A/B SWM, the Switching Matrix Subsystem consists of one HP 4089A/B SWM and two HP 4084B SWCs. One SWC controls measurement pins 1 to 48 (block 1), and the other SWC controls measurement pins 49 to 96 (block 2).

All pin-port interconnections are controlled by the input relays for each block. Input relays are automatically controlled by the Connect_pin functions, which connect the specified pin(s) and port(s). Refer to the HP 4089A/B SWM Diagram discussion in chapter 2 of the *System Information* Manual for more information.

Connect_pin

Note



If you connect a measurement port to measurement pins in both blocks 1 and 2, guard capacitance increases more than usual, resulting in the following:

- For capacitance measurements, the accuracy of the measurement results is not guaranteed. Capacitance measurement error correction is valid only when the CML or CMH ports of the CMS are connected to two measurement pins in either block 1 or block 2.
 - For SMU measurements, SMU oscillation may occur.
-

Example

```
. . . .  
int err;  
. . . .  
err=connect_pin(fnsmu(1), 32);  
err=connect_pin2(fnsmu(1), 12,16);  
err=connect_pin3(fngnd(), 28, 32, 36);  
. . . .  
. . . .
```

See Also

Connect_th
Disconnect_all

Connect_th

This program connects a specified port to a range of measurement pins.

Synopsis

```
#include <pcs/tis.h>

int connect_th(port, pin_top, pin_tail)
int port, pin_top, pin_tail;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address to be connected to the specified <i>pin</i> . 32701 to 32711 and 32721 to 32732 ¹
<i>pin_top, pin_tail</i>	Specify a range of pin numbers. The <i>pin_tail</i> value must be greater than the <i>pin_top</i> value. 1 through 96

¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems.

Description

This function connects a range of pins to one port. All available pins specified from *pin_top* to *pin_tail* are connected to the specified *port*. No more than one port can be connected to a measurement pin. Thus, if the same *pin number* appears in two or more Connect_pin functions that specify different ports, only the port specified in the first connect_pin function is connected to the measurement pin.

A value returned from Fnport, Fnsmu, Fnaux, Fncmh, Fncml, Fngnd, Fngsmu, or Fngcmu is suitable for the *port* argument.

With the exception of connecting a range of pins to one port, Connect_th is functionally identical to the Connect_pin function. Refer to the description of the Connect_pin function for further details.

Connect_th

Example

```
. . .  
int err;  
. . .  
err=connect_th(fnsmu(3), 37, 48);  
. . .
```

See Also

Connect_pin
Disconnect_all

Disconnect_all

This function disconnects all SWM pins from all ports.

Synopsis

```
#include <pcs/tis.h>

int disconnect_all()
```

Description

This function disconnects all SWM pins from all ports and sets the SWM to the initialized status. The function is the equivalent of calling the `Connect_pin(0,0)` function, but is provided for better program readability.

Note



If you connect the terminals of unsupported instruments to the AUX ports, use dry-switching. Before executing any `Connect_pin` function, confirm that unsupported instruments force 0 V.

Example

```
. . .
int err;
. . .
err=disconnect_all();
. . .
```

See Also

`Connect_pin`

General I-V Measurements

Introduction

The C Library provides the following functions for general source/monitor control and measurements:

- Disable_port
- Disable_port_all
- Disable_port2
- Disable_port3
- Force_i
- Force_meas
- Force_v
- Long_cal
- Measure_i
- Measure_p
- Measure_v
- Port_status
- Set_pbias
- Set_piv
- Set_smu
- Set_vm
- Short_cal

These functions are controls for basic dc current and voltage measurements. Use the functions to set the current or voltage output and to make the measurements.

Disable_port, Disable_port2, and Disable_port3, are described collectively in this chapter under the Disable_port heading.

The functions are arranged alphabetically in the chapter.

Disable_port

These functions set the specified ports, measurement pins, or units to zero output status.

Synopsis

```
#include <pcs/tis.h>

int disable_port(port)
int port;

int disable_port2(port1, port2)
int port1, port2;

int disable_port3(port1, port2, port3)
int port1, port2, port3;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number currently connected to a port, or a DCS, CMU, or CMU84 unit address to be disabled. <i>port number</i> 0, 32701 to 32706, 32708 to 32710 ¹ , 32721 to 32732 ² <i>unit address</i> 0, 32621 to 32633, 32635, 32637, 32639 . . . 32663, 32671, 32676 <i>pin number</i> 0, 1 to 96 ³

¹ Only ports connected to the SWM can be specified. 32710 is available only when an SMU is connected to the SMU1 port.

² If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

³ 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

Description

To disable two or more pins or ports, use `Disable_port2`, `Disable_port3`, or `Disable_port_all`.

Port is used to specify units that are connected to pins or ports, and are to be disabled. A unit address can be used to specify units that are not connected to the SWM, and are to be disabled.

If 0 is specified, all ports are set to zero output when this function is executed.

Example

```
. . .  
. . .  
if (disable_port(12) == -1) error_rep();  
. . .  
. . .
```

See Also

Disable_port_all

Disable_port_all

This function sets every port that has voltage or current force ability to the zero output status.

Synopsis

```
#include <pcs/tis.h>

int disable_port_all()
```

Description

All ports and units are set to zero output status when this function is executed.

If you want to disable one, two or three pins or ports, use `Disable_port`, `Disable_port2`, or `Disable_port3`.

Example

```
. . .
. . .
if (disable_port_all() == -1) error_rep();
. . .
. . .
```

See Also

`Disable_port`
`Disable_port2`
`Disable_port3`

Force_i

This function sets the output current of a specified SMU.

Synopsis

```
#include <pcs/tis.h>

int force_i(port, current, range, compliance)
int port;
double current, range, compliance;
```

Arguments

Item	Range Restriction/Description						
<i>port</i>	Specifies an SWM port address (returned by Fnport or Fnsmu), a pin number currently connected to a port, or a DCS unit address from which the specified current is being forced. <table> <tr> <td><i>port address</i></td><td>32701 to 32704, 32729, 32732 ¹</td></tr> <tr> <td><i>unit address</i></td><td>32621 to 32632</td></tr> <tr> <td><i>pin number</i></td><td>1 to 96 ²</td></tr> </table>	<i>port address</i>	32701 to 32704, 32729, 32732 ¹	<i>unit address</i>	32621 to 32632	<i>pin number</i>	1 to 96 ²
<i>port address</i>	32701 to 32704, 32729, 32732 ¹						
<i>unit address</i>	32621 to 32632						
<i>pin number</i>	1 to 96 ²						
<i>current</i>	A current value to be forced. Numeric expression [A]: -1 to 1 ³						
<i>range</i>	A current output range. 0.0 for Auto range OR: Numeric expression [A]: -1 to 1 ³						
<i>compliance</i>	Specifies a voltage compliance value. REMAIN macro OR: Numeric expression [V]: -200 to 200 ⁴						

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ For details on range for *current* and *range*, refer to the Description.

⁴ Range restrictions for *compliance* depend on the specified *current*. Refer to Description for details.

⁵ If *current* is larger than 100 mA, default value is set to 2 V.

Force_i

Description

An SMU port address, unit address, or pin number of any pin connected to the SMU can be used to specify the SMU *port*.

The actual range restrictions of *current*, *range*, and *compliance* depend on the SMU and SMU port used.

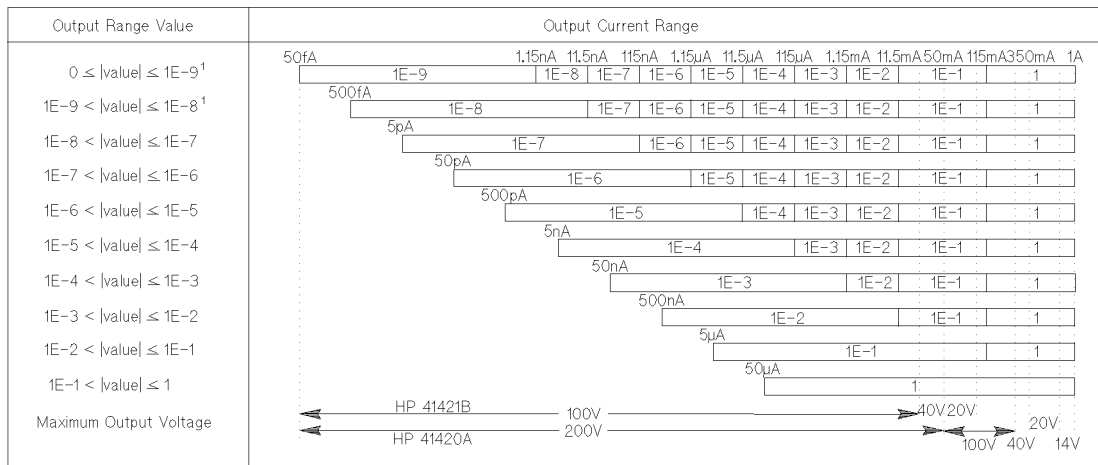
The *range* argument sets the current range of the SMU. An exact *range* value is not necessary. Specify an arbitrary value within the current range allowed and the system chooses the appropriate one.

If you specify 0 for *range*, the SMU is set to Auto range mode. If you specify a range, however, the SMU is set to Limited Auto range mode.

The output current resolution and the range you can set for *compliance* (maximum output voltage) depend on the specified *current*. If the specified *current* is high, the maximum allowable voltage compliance and output current resolution drop accordingly.

You can use the REMAIN macro for the *compliance* argument. REMAIN specifies that the compliance remain at the present setting unless the unit changes voltage or current source mode by this function. If the specified unit changes voltage or current source status, specifying the REMAIN macro sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

Setting the *range* to a low value improves output current resolution but may increase settling time because of additional ranging time and wait time. Refer to the following table for the relationship between output current, range, and voltage compliance.



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

The maximum output voltage of an SMU operating as a current source (IS mode) is set by the specified *compliance*. Voltage compliance should not be set too high, because the SMU may output a pulse when changing range.

5-6 General I-V Measurements

Example

```
. . .  
if (force_i(drain, i_ds1, i_ds2, 2.0) == -1) error_rep();  
. . .
```

See Also

Force_meas

Force_v

Force_meas

This function sets the output current or voltage of the specified SMU or VS, measures the dc current or voltage using the specified SMU or VM, and returns the measurement value.

Synopsis

```
#include <pcs/tis.h>

int force_meas(source_port, measure_port, source_mode,
               source, measure, source_range, measure_range,
               compliance, hold_time)
int source_port, measure_port, source_mode;
double source, *measure, source_range, measure_range;
double compliance, hold_time;
```

Arguments

Item	Range Restrictions/Description
<i>source_port</i>	Specifies an SWM port address (returned by the Fnport, Fnsmu, or FnauX), an SWM pin number that is connected to a port, or a DCS unit address (returned by Fnunitsmu or Fnvs) <i>port address</i> 32701 to 32709, 32721 to 32732 ¹ <i>unit address</i> 32621 to 32633, 32635, 32637, . . . , 32663 <i>pin number</i> 1 to 96 ²
<i>measure_port</i>	Specifies an SWM port address (returned by the Fnport, Fnsmu, or FnauX), an SWM pin number that is connected to a port, or a DCS unit address (returned by Fnunitsmu or Fnvm). <i>port address</i> 32701 to 32709, 32721 to 32732 ¹ <i>unit address</i> 32621 to 32632, 32634, 32636, . . . , 32664 <i>pin number</i> 1 to 96 ²
<i>source_mode</i>	Determines the voltage or current source mode for <i>source_port</i> . You may choose from one of the two macros below: V_SOURCE (= 1) L_SOURCE (= 2)
<i>source</i>	A voltage value when the <i>source_mode</i> is set to V_SOURCE and a current value when the <i>source_mode</i> is set to L_SOURCE mode. The voltage or current is forced to the specified <i>source_port</i> .
<i>measure</i>	A pointer to a double where the measurement value is stored.

Item	Range Restrictions/Description
<i>source_range</i>	Specifies a forcing range. 0.0 for Auto range mode OR: When using SMU in VS mode [V]: -200 to 200 ³ When using VS [V]: -40 to 40 When using SMU in IS mode [A]: -1 to 1 ⁴
<i>measure_range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: When using SMU in VS mode [A]: -1 to 1 When using VM [V]: -40 to 40 When using VS, or an SMU in IS mode:— ⁵
<i>compliance</i>	Specifies a voltage or current compliance according to <i>source_mode</i> . REMAIN macro OR: If the SMU is in VS mode [A]: -1 to 1 ⁴ If VS is used:— ⁵ If the SMU is in IS mode [V]: -200 to 200 ³
<i>hold_time</i>	The HP 4142B waits this amount of time to start measurements after the voltage or current force has been set. 0 to 99.9999 Resolution: 0.0001

- ¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.
- ² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.
- ³ Maximum output voltage and actual range restrictions for *compliance* depend on the specified voltage range, and SMU and SMU port used.
- ⁴ Maximum output current and actual range restrictions for *compliance* depend on the specified test current, and SMU and SMU port used.
- ⁵ This is a dummy parameter.
- ⁶ If the *source* is larger than 100 mA, default value is set to 2 V.

Force_meas

Description

When using an SMU (other than the source port or source unit) as a measurement port or measurement unit, the SMU must be set to the measurement mode required to perform the desired measurement with *source_mode*. Use I_SOURCE for current source and voltage measurements. Use V_SOURCE for voltage source and current monitor. You can make this setting with Force_i or Force_v.

The maximum output value, resolution, and output compliance depend on the specified *source_range* and *measure_range*. Refer to the Force_v or Force_i functions for details.

Hold_time is the time between voltage or current forcing and when the measurement is made.

You can use the REMAIN macro for the *compliance* argument. REMAIN specifies that the compliance remain at the present setting unless the unit changes voltage or current source mode by this function. If the specified unit changes voltage or current source status, specifying the REMAIN macro sets the current compliance to 100 μ A or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

In Offline Mode:

This function returns simulated measurement values to the *measure* argument. The type of value returned depends on the simulation mode: constant data or file data simulation mode.

Constant Data Simulation mode:

Returned values are determined by the *measure_range* value as follows:

For current measurements:

Unit	Meas. Range	Returned Value	Polarity
SMU	= 0	I compliance	Same as SMU <i>source</i>
	\neq 0	Range Value	
VS	any	20 mA	Same as VS <i>source</i>

For voltage measurements:

Unit	Meas.Range	Returned Value	Polarity
SMU	—	V compliance	Same as SMU <i>source</i>
VM	= 0	2.0 V	Always positive
	\neq 0	Range Value	Same as <i>measure_range</i>

If you do not set a *measure_range* for the SMUs, this function uses the last valid range setting as the measurement range value.

File Data Simulation mode:

This function reads <meas> from the data simulation file and returns this value. The data simulation file format is as follows:

```
Force_meas <meas>
```

See Also

Force_i
Force_v
Measure_i
Measure_v

Force_v

This function sets the output voltage for a specified SMU or VS of the DCS, and sets the internal dc bias source of the CMU.

Synopsis

```
#include <pcs/tis.h>

int force_v(port, voltage, range, compliance)
int port;
double voltage, range, compliance;
```

For DCS Measurements:

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address (returned by Fnport, Fnaux, or Fnsmu), a pin number currently connected to a port, or a DCS unit address from which the specified voltage is being forced. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>voltage</i>	Specifies a voltage value to be forced. Numeric expression [V]: -200 to 200 ³
<i>range</i>	Specifies an output voltage range. 0.0 for Auto range mode OR: Numeric expression [V]: -200 to 200 ³
<i>compliance</i>	Specifies a current compliance value. REMAIN macro OR: Numeric expression [V]: -1 to 1 ³

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ Maximum output voltage and the actual range restrictions for current *compliance* depend on the specified voltage range. Refer to Description for details.

For VSs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address (returned by Fnport or Fnsmu), a pin number currently connected to a port, or a DCS unit address from which the specified voltage is being forced. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32633, 32635, 32637 . . . 32663 <i>pin number</i> 1 to 96 ²
<i>voltage</i>	Specifies a voltage value to be forced. Numeric expression [V]: -40 to 40
<i>range</i>	Specifies an output voltage range. Numeric expression [V]: -40 to 40 Recommended range: 0, 20, 40
<i>compliance</i>	Specifies a current compliance value. Numeric expression:— ³

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VS port.

³ This is a dummy parameter.

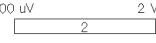
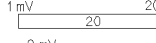
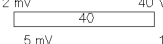
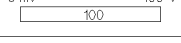
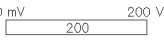
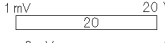
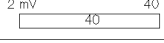
Description

A port address, unit address, or pin number of any measurement pin connected to the port can be used to specify the SMU or VS *port*. If an SWM port or pin number is passed, the associated unit should be an SMU or a VS.

The actual range restrictions of the *voltage*, *range*, and *compliance* arguments of the SMU depend on the SMU and SMU port used. It is not necessary to set an exact *range* value for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

The voltage range of the SMU is set by the output *range* argument. The maximum output voltage, voltage resolution, and the range of current compliance (maximum output current) depend on the specified voltage range. VSs do not have a current compliance setting.

Force_v

Unit	Output Range Value	Output Voltage Range	Maximum Output Current	
HP 41420A HP 41421B	0	Depends on the previous setting ¹		
	$0 \leq value \leq 2$		HP 41420A 1 A	HP 41421B 100 mA
	$2 < value \leq 20$		1 A (≤ 14 V) 700 mA (> 14 V)	100 mA
	$20 < value \leq 40$		350 mA	50 mA
	$40 < value \leq 100$		125 mA	20 mA
HP 41420A	$100 < value \leq 200$		50 mA	
VS (HP 41424A)	0	Depends on the previous setting		
	$0 < value \leq 20$		100 mA	
	$20 < value \leq 40$		200 mA	

¹ If the previous setting of the SMU was IS mode, the optimum voltage range is automatically selected for the specified output voltage. However, if the SMU is already set to VS mode, the voltage range does not change, regardless of the specified output voltage, and if the specified output voltage is outside the limits of the range, TIS ERROR 56 is reported.

You can use the **REMAIN** macro for the SMU *compliance*. **REMAIN** specifies that the compliance remain at the present setting unless the unit changes voltage or current source mode by this function. If the specified unit changes voltage or current source status, specifying the **REMAIN** macro sets the current compliance to 100 μ A or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

For CMU and CMU84 Measurements:

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address (returned by Fnport, Fnaux, or Fncmh) a pin number currently connected to a port, a CMU or CMU84 unit address (returned by the Fncmu or Fncmu84 function). <i>port address</i> 32708, 32727 ¹ <i>unit address</i> 32671 (for CMU), 32676 (for CMU84) <i>pin number</i> 1 to 96 ²
<i>voltage</i>	A voltage value to be forced. Numeric expression [V]: -100 to 100 (for CMU) ³ Numeric expression [V]: -40 to 40 (for CMU84)
<i>range</i>	An output voltage range. 0.0 for Auto range mode OR: Numeric expression [V]: -100 to 100 (for CMU) ³ Numeric expression [V]: -40 to 40 (for CMU84)
<i>compliance</i>	Numeric expression:— ⁴

¹ Must be the CMH and CML terminals of the CMU connected to AUX3 (CMH) and AUX4 (CML) ports, respectively, or connected to AUX31 and AUX41 ports, respectively, if the SWM is equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the CMH port.

³ Actual output *voltage* range restrictions depend on the specified bias range.

⁴ This is a dummy parameter.

Description

For CMU Measurements:

A CMH port address, CMH unit address, or a pin number of any pin connected to the CMH port can be used to specify the CMH *port*. If an SWM port or pin number is passed, the associated unit should be a high terminal of the CMU or CMU84.

The *range* sets the output range of the CMU internal dc bias source. It also determines the maximum output voltage and output resolution. Refer to the following table:

Force_v

Output Range Value	DC Bias Range	Max. Output Voltage	Max. Resolution	Max. Output Current
$0 < \text{value} \leq 1.999$	1 V	± 1.999 V	1 mV	± 5 mA
$1.999 < \text{value} \leq 19.999$	10 V	± 19.99 V	10 mV	± 5 mA
$19.99 < \text{value} \leq 100$	100 V	± 100.0 V	100 mV	± 5 mA
0	last valid range setting			

It is not necessary to set an exact *range* value for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

If the output current exceeds approximately 5 mA, the V OUTPUT indicator on the CMU blinks and the internal dc bias source is not able to output the specified voltage.

The *compliance* argument is always ignored in this measurement.

For CMU84 Measurements:

A CMH port address, CMH84 unit address, or a pin number of any pin connected to the CMH port can be used to specify the CMH.

Output *voltage* defines the dc bias voltage.

If Option 321/531 (HP 4284A Option 001) is *not* installed, the dc bias voltage can be set to 0 V, 1.5 V, or 2.0 V, as shown in the following table:

Output <i>voltage</i> Value	Actual Dc Bias Voltage
$0 \leq \text{value} < 1.5$	0 V
$1.5 \leq \text{value} < 2.0$	± 1.5 V
$ \text{value} = 2.0$	± 2.0 V
$2.0 < \text{value} $	error

If Option 321/531 (HP 4284A Option 001) is installed, the allowable dc bias voltage is 0 through ± 40.0 V, as shown in the following table:

Output <i>voltage</i> Value	Actual Dc Bias Voltage	Resolution
$0 \leq \text{value} \leq 4.000$	value	1 mV
$4.000 < \text{value} \leq 8.000$	value	2 mV
$8.000 < \text{value} \leq 20.000$	value	5 mV
$20.000 < \text{value} \leq 40.00$	value	10 mV
$40.00 < \text{value} < 40.01$	± 40.00 V	—
$40.01 \leq \text{value} $	error	—

Example

```
. . .
int err;
. . .
err = force_v(fncmh(), 5.0, 20.0, 1E-3);
err = measure_cmu(&capacitance, &conductance, 0.0);
. . .
```

See Also

Force_i
Force_meas

Long_cal

This function sets all SMUs, VSs, VMs, and the AFU to their initial statuses. It also initiates the automatic calibration function of the DCS, which measures and compensates for the dc offset voltages of the SMUs, VSs, VMs, AFU, A/D and D/A converter.

Synopsis

```
#include <pcs/tis.h>

int long_cal()
```

Description

This function puts the SMUs and VSs of the DCS in their initial states, and calibrates for approximately 21 seconds for a standard system configuration.

You should perform Long_cal after your system has fully warmed up to correct for offsets caused by temperature drifts.

For best measurement results, perform Short_cal every 30 minutes or when the ambient temperature changes more than 3°C (6°F). Calibration is performed at about 30 minute intervals if all DCS (HP 4142B) units are in a disabled state for longer than 30 minutes.

Example

```
. . .
if (long_cal()==-1) error_rep();
. . .
```

See Also

Short_cal

Measure_i

This function measures dc current using a specified SMU or VS and returns the measurement value in amperes.

Synopsis

```
#include <pcs/tis.h>

int measure_i(port, current, range)
int port;
double *current, range;
```

Arguments

Item	Range Restrictions/Description						
<i>port</i>	Specifies an SWM port address (returned by Fnport, Fnsmu, or Fnaux), an SWM pin number currently connected to a port, or a DCS unit address (returned by Fnunitsmu or Fnvs). <table> <tr> <td><i>port address</i></td><td>32701 to 32706, 32708, 32709, 32721 to 32732 ¹</td></tr> <tr> <td><i>unit address</i></td><td>32621 to 32633, 32635, 32637, 32639 . . . 32663</td></tr> <tr> <td><i>pin number</i></td><td>1 to 96 ²</td></tr> </table>	<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹	<i>unit address</i>	32621 to 32633, 32635, 32637, 32639 . . . 32663	<i>pin number</i>	1 to 96 ²
<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹						
<i>unit address</i>	32621 to 32633, 32635, 32637, 32639 . . . 32663						
<i>pin number</i>	1 to 96 ²						
<i>current</i>	A pointer to a double variable where the measurement result is stored. Any valid pointer						
<i>range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: When using SMU in VS mode: -1 to 1 ³ When using VS: this is a dummy parameter						

¹ The port specified by a port address of 32705, 32706, 32708, 32709, 32721 to 32728, 32730, or 32731 must be connected to a VS. If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the SMU or VS that will perform the measurement.

³ Actual range restrictions for measurement *range* depend on the SMU and SMU port used. Refer to Description for details.

Measure_i

Description

A port address, unit address, or pin number of any measurement pin connected to the SMU or VS port can be used to specify the SMU or VS *port*. If an SWM port or pin number is used, the associated unit should be an SMU or VS. When using an SMU, the SMU specified must be set to the VS mode. The current range of the SMU is set with *range*, and the measurement value is returned to *current*.

The current range of the VS depends on its voltage range. For the 20 V range, the 100 mA range is used; for the 40 V range, the 20 mA range is used. It is not necessary for the exact range to be defined for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

If you specify 0 for the measurement *range* of the SMU, it is set to the Auto range mode. If you specify a range, however, the SMU is set to the Limited Auto range mode.

Measurement Range Value	Measurement Range and Current Range										
$0 \leq \text{value} \leq 1\text{E-}9^1$	20fA	1E-9	1.15nA	11.5nA	115nA	1.15uA	11.5uA	115uA	1.15mA	115mA	1A
$1\text{E-}9 < \text{value} \leq 1\text{E-}8^1$	200fA	1E-8	1E-8	1E-7	1E-6	1E-5	1E-4	1E-3	1E-2	1E-1	1
$1\text{E-}8 < \text{value} \leq 1\text{E-}7$	2pA	1E-7	1E-7	1E-6	1E-5	1E-4	1E-3	1E-2	1E-1	1	
$1\text{E-}7 < \text{value} \leq 1\text{E-}6$	20pA	1E-6	1E-6	1E-5	1E-4	1E-3	1E-2	1E-1	1		
$1\text{E-}6 < \text{value} \leq 1\text{E-}5$	200pA	1E-5	1E-5	1E-4	1E-3	1E-2	1E-1	1			
$1\text{E-}5 < \text{value} \leq 1\text{E-}4$	2nA	1E-4	1E-4	1E-3	1E-2	1E-1	1				
$1\text{E-}4 < \text{value} \leq 1\text{E-}3$	20nA	1E-3	1E-3	1E-2	1E-1	1					
$1\text{E-}3 < \text{value} \leq 1\text{E-}2$	200nA	1E-2	1E-2	1E-1	1						
$1\text{E-}2 < \text{value} \leq 1\text{E-}1$	2uA	1E-1	1E-1	1							
$1\text{E-}1 < \text{value} \leq 1$	20uA	1	1								

← HP 41421A HP 41420A →

¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

Setting the *range* of the SMU to a low value improves measurement resolution but may increase measurement time because of additional ranging and wait time.

In Offline Mode:

This function returns simulated measurement values to the *current* argument. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

Returned values are determined by the *range* value as follows:

Unit	Meas.Range	Returned Value	Polarity
	0	I compliance	
SMU	$\neq 0$	Range Value	Same as SMU output value
VS	any	20 mA	Same as VS output value

File Data Simulation mode:

This function reads <meas> from the data simulation file and returns this value. The data simulation file format is as follows:

```
Measure_i      <meas>
```

Example

```
. . .
if (measure_i(12, &current, 1E-3) == -1) error_rep();
. . .
```

See Also

Force_meas
Measure_v

Measure_p

This function performs pulsed spot measurements according to the conditions established by the Set_pbias function, then returns the measurement values.

Synopsis

```
#include <pcs/tis.h>

int measure_p(port, mode, value, range)
int port, mode;
double *value, range;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port number, a pin number, or a unit address. <i>port address</i> 32701 to 32706, 32708, 32709, 32721 to 32732 ⁴ <i>unit address</i> 32621 to 32632, 32634, 32636, 32638, . . . , 32664 <i>pin number</i> 1 to 96 ¹
<i>mode</i>	Specifies either voltage or current measurement mode at the <i>port</i> . Use one of the following macros: V_MEAS (= 1) I_MEAS (= 2)
<i>value</i>	Specifies a pointer to a double where the measurement result is stored.
<i>range</i>	Specifies a measurement range. 0.0 for Auto range OR: For SMU voltage measurements [V]:— ² For SMU current measurements [A]: −1 to 1 ³ For VM voltage measurements [V]: −40 to 40

¹ 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the SMU or VS that performs the measurement.

² This is a dummy parameter. The voltage range of the SMU depends on the voltage compliance value.

³ Actual range restrictions for the measurement *range* depend on the SMU and SMU port used. If zero is specified for the current range, the current range is set to the range that can measure current compliance with the highest resolution.

⁴ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

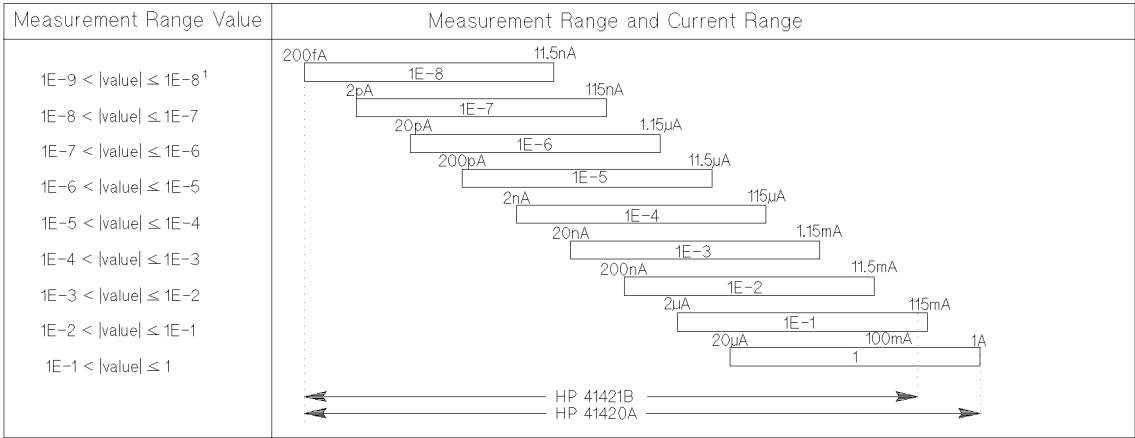
Description

A port address, unit address, or the pin number of any measurement pin connected to the SMU or VM port can be used to specify the SMU or VM *port*. The specified unit must be set to desired mode (IS mode for voltage measurements; VS mode for current measurements).

The current range of an SMU and the voltage range of a VM are determined by the measurement *range*. The voltage range of an SMU is determined by Set_pbias or Force_i voltage compliance. When performing pulsed measurements, differential voltage measurements cannot be performed.

It is not necessary for the exact range value to be defined for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

Measurement Mode = I_MEAS (current measurements):



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

Measure_p

Measurement Mode = V_MEAS (voltage measurements):

Unit	Voltage Compliance Setting	Measurement Range
HP 41420A	$0 \leq value \leq 2$	40 μ V 2 V 2
HP 41421B	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40
	$40 < value \leq 100$	2 mV 100 V 100
HP 41420A	$100 < value \leq 200$	4 mV 200 V 200
VM (HP 41424A)	Measurement Range Value	800 μ V 40 V
	0	40
	$0 < value \leq 2$	40 μ V 2 V 2
	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40

In Offline Mode:

This function returns simulated measurement values to the measurement value *name* argument. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

Returned values are determined by the measurement *range* value as follows:

For current measurements:

Unit	Meas.Range	Returned Value	Polarity
SMU	= 0	I compliance	Same as SMU output value
	$\neq 0$	Range Value	
VS	any	20 mA	Same as VS output value

For voltage measurements:

Unit	Meas.Range	Returned Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	$\neq 0$	Range Value	Same as measurement <i>range</i>

File Data Simulation mode:

This function reads <meas> from the data simulation file and returns this value. The data simulation file format is as follows:

```
Measure_p      <meas>
```

Example

```
. . .  
int err;  
double current;  
. . .  
err = set_pbias(12, V_SOURCE, 20, -3, 15, 0.4, 1, 0.5, 1E-3);  
err = measure_p(12, I_MEAS, &current, 1E-3);  
. . .  
. . .
```

See Also

Set_pbias
Set_piv

Measure_v

The function measures dc voltage using the specified SMU or VM and returns the measurement value in volts (V).

Synopsis

```
#include <pcs/tis.h>

int measure_v(port, voltage, range)
int port;
double *voltage, range;
```

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, an SWM pin number, or a unit address. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>voltage</i>	A pointer to a double variable where the measurement result is stored. Any valid pointer
<i>range</i>	Specifies a measurement range. Dummy parameter for this measurement.

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the SMU that performs the measurement.

For VMs:

Item	Range Restrictions/Description
<i>port</i>	Specifies a DCS port address, a unit address, or a pin number. <i>AUX port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>AUX unit address</i> 32634, 32636, 32638 . . . 32664 <i>pin number</i> 1 to 96 ²
<i>voltage</i>	A pointer to a double variable where measurement results are stored. Any valid pointer
<i>range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: –40 to 40

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VM port.

³ VM measurement mode is specified by the Set_vm function.

Description

A port address, unit address, or a pin number of any measurement pin connected to the port can be used to specify the SMU or VM *port*. The specified SMU must be set to IS mode. If an SWM port or pin number is used, the associated unit should be an SMU or VM. The measurement value is returned to *voltage*.

The voltage range of the SMU is set by the *compliance* argument specified in the Force_i function. The voltage range of the VM is set by *range*. It is not necessary for the exact range to be defined for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

The 0.2 V range is available only when a VM is set to the differential voltage measurement mode by the Set_vm function.

Measure_v

**VM Voltage Ranges for Grounded Measurement Mode
and SMU Voltage Range**

Unit	Voltage Compliance Setting	Measurement Range
HP 41420A	$0 \leq value \leq 2$	40 μ V 2 V 2
HP 41421B	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40
	$40 < value \leq 100$	2 mV 100 V 100
HP 41420A	$100 < value \leq 200$	4 mV 200 V 200
VM (HP 41424A)	Measurement Range Value	40 μ V 2 V 20 V 40 V
	0	2 20 40
	$0 < value \leq 2$	40 μ V 2 V 2
	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40

VM Voltage Ranges for Differential Measurement Mode

Unit	Measurement Range Value	Measurement Range and Current Range	Maximum Voltage (VM terminal–Circuit Common)
VM (HP 41424A)	0	4 μ V 0.2 V 2 V 4 μ V 0.2 V	Max. 40 V
	$0 < value \leq 0.2$	40 μ V 2 V	
	$0.2 < value \leq 2$		

In Offline Mode:

This function returns simulated measurement values to *voltage*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

Returned values are determined by the measurement *range* value as follows:

Unit	Meas.Range	Returned Value	Polarity
SMU	—	V compliance	Same as SMU <i>output value</i>
VM	= 0	2.0 V	Always positive
	$\neq 0$	Range Value	Same as <i>measurement range</i>

If you do not set a measurement range for SMUs, this function uses the last valid range setting as the *range* value.

File Data Simulation mode:

This function reads <meas> from the data simulation file and returns this value. The data simulation file format is as follows:

```
Measure_v      <meas>
```

Example

```
. . .  
if (measure_v(drain, &v_th1, 1.8) == -1) error_rep();  
. . .
```

See Also

Force_meas
Measure_i

Port_status

This function returns the status of the specified port when a measurement is done.

Synopsis

```
#include <pcs/tis.h>

int port_status(port, status)
int port, *status;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or a unit address.
	<i>port address</i> 32701 to 32732 ¹
	<i>unit address</i> 32621 to 32665, 32676, 32677, 32671, 32672
	<i>pin number</i> 1 to 96 ²
<i>status</i>	A pointer to an integer where the status code is stored. The status code returned is one of the following: NORMAL_MEAS (= 0) DCS_COMP_OTHER (= 1) DCS_COMP (= 2) DCS_OSC (= 3) DCS_OVERFLOW (= 4) DCS_SWP_STOPPED (= 5) CMS_OVERFLOW (= 6) CMU84_UNBALANCE (= 1) CMU84_ADC_DOWN (= 2) CMU84_OVERLOAD (= 3) CMU84_ALC_DOWN (= 4)

¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

Description

A pin number, unit address, or port address can be used to specify the *port* (or ports) where the status is to be returned. The values returned to *status* are predefined port status as shown in the following table:

Unit Name	Status Macro: Number	Condition
SMU, VS, VM	NORMAL_MEAS: 0	Normal measurement
	DCS_COMP_OTHER: 1	Another unit reached compliance or current limit.
	DCS_COMP: 2	The SMU reached compliance or the VS reached current limit.
	DCS_OSC: 3	An SMU is oscillating.
	DCS_OVERFLOW: 4	A dc source measurement overflow.
GNDU	NORMAL_MEAS: 0	Normal measurement
CMU	NORMAL_MEAS: 0	Normal measurement
	CMS_OVERFLOW: 4	A C meter measurement overflow.
CMU84	CMU84_UNBALANCE: 1	Analog bridge is unbalanced.
	CMU84_ADC_DOWN: 2	A/D converter is not working.
	CMU84_OVERLOAD: 3	Signal source is overloaded.
	CMU84_ALC_DOWN: 4	Automatic Level Control (ALC) is unable to be regulated.

Port_status

The relationship between port name, number, and unit name is shown in this table.

Port Name	Port Number	Unit Name
SMU1	1	SMU1
GSMU	10, -1	
SMU2	2	SMU2
SMU3	3	SMU3
SMU4	4	SMU4
SMU5	29	SMU5
SMU6	32	SMU6
AUX1 ¹	5	VS, VM
AUX2 ¹	6	
AUX3 ¹	8	
AUX11 ¹	21	
AUX12 ¹	22	
AUX13 ¹	23	
AUX21 ¹	24	
AUX22 ¹	25	
AUX23 ¹	26	
AUX31 ¹	27	
AUX32 ¹	28	
AUX41 ¹	30	
AUX42 ¹	31	
GNDU	7	GNDU
AUX3(CMH) ¹	8	CMU
AUX4(CML) ¹	9	
AUX31(CMH) ¹	27	
AUX41(CML) ¹	30	
GCMU ¹	11, -8, -9	

¹ These ports report the status of the unit actually connected to the SWM port.

The following table shows the relationship between unit address and unit.

Unit Address	Unit
32621 to 32628	HP 41421B (SMU 100 V/100 mA)
32629 to 32632	HP 41420A (SMU 200 V/1 A)
31633, 32635, 32637, 32639, 32641, 32643, 32645, 32647, 32649, 32651, 32653, 32655, 32657, 32659, 32661, 32663	HP 41424A (VS)
31634, 32636, 32638, 32640, 32642, 32644, 32646, 32648, 32650, 32652, 32654, 32656, 32658, 32660, 32662, 32664	HP 41424A (VM)
32671	CMU (CMH)
32672	CMU (CML)
32676	CMU84 (CMH)
32677	CMU84 (CML)

The returned status of any unused port is always DCS_NORMAL. If two or more conditions occur during a measurement, the priority of the returned code is the following order:

DCS_OSC > DCS_OVERFLOW > DCS_COMP > DCS_COMP_OTHER

For instance, if an SMU reaches compliance and is oscillating at the same time, a DCS_OSC (oscillating) is returned to *status*. The *status* condition remains unchanged until the next measurement.

In Offline Mode:

This function returns simulated measurement values to the *status* argument in the offline mode. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns 0 to *status* for all measurements.

File Data Simulation mode:

This function reads <status> from the data simulation file and returns the value. The data simulation file format is as follows:

Port_status <status>

Example

```
. . .
if (port_status(fnsmu(1), &smu_status) == -1) error_rep();
. . .
```

Set_pbias

This function sets the parameters for a pulsed spot measurement or a sweep measurement with pulsed bias using Measure_p and Sweep_iv, respectively.

Synopsis

```
#include <pcs/tis.h>

int set_pbias(port, mode, range, base, peak, width, period,
              hold, compliance)
int port, mode;
double base, peak, width, period, hold, compliance;
```


Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or a unit address. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies either the voltage or current force mode. Choose from the following macros: V_SOURCE (= 1) I_SOURCE (= 2)
<i>range</i>	Specifies a voltage or current force range. 0.0 for Auto range mode OR: For voltage source: numeric expression [V]: -200 to 200 ³ For current source: numeric expression [A]: -1 to 1
<i>base, peak</i>	Specifies the pulse base and peak values of the voltage or current being forced. These values must have the same polarity to force current. For voltage source: numeric expression [V]: -200 to 200 ³ For current source: numeric expression [A]: -1 to 1
<i>width, period</i>	Specifies the pulse timing parameters. Specify 0.0 for 0.01 second <i>period</i> OR: Numeric expression [s]: 0.01 to 0.05 ^{4, 5} Resolution: 1E-4
<i>hold</i>	Once the pulsed measurement is triggered, the dc source waits the number of seconds defined by this argument before raising the edge of a pulse. Numeric expression [s]: 0 to 655.35 Resolution: 0.01
<i>compliance</i>	Specifies the voltage or current compliance on the <i>port</i> . REMAIN macro OR: For voltage source: numeric expression [A]: -1 to 1 ³ For current source: numeric expression [V]: -200 to 200

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ Actual range restriction of this value depends on the unit used. The values of *base*, *peak*, and *compliance* depend on the specified voltage range.

⁴ The *width* must be less than 50% of the *period*.

⁵ If the *period* is set to “0”, the pulse period is not set.

Set_pbias

When using a VS as a voltage pulse source:

Item	Range Restrictions/Description
<i>port</i>	Specifies a port address, a pin number, or a unit address. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32633, 32635, 32637, 32639 . . . 32663 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies either the voltage or current force mode. Choose from the following macros: V_SOURCE (= 1) I_SOURCE (= 0)
<i>range</i>	Specifies a voltage or current force range. 0.0 for Auto range mode OR: Numeric expression [V]: -40 to 40
<i>base, peak</i>	Specifies the pulse base and peak values of the voltage or current being forced. Numeric expression [V]: -40 to 40
<i>width, period</i>	Specifies the pulse timing parameters. Specify 0.0 for 0.01 second <i>period</i> OR: Numeric expression [s]: 0.01 to 0.5 ^{3, 4} Resolution: 1E-4
<i>hold</i>	Once the pulsed measurement is triggered, the dc source waits the number of seconds defined by this argument before raising the edge of a pulse. Numeric expression [s]: 0 to 655.35 Resolution: 0.01

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

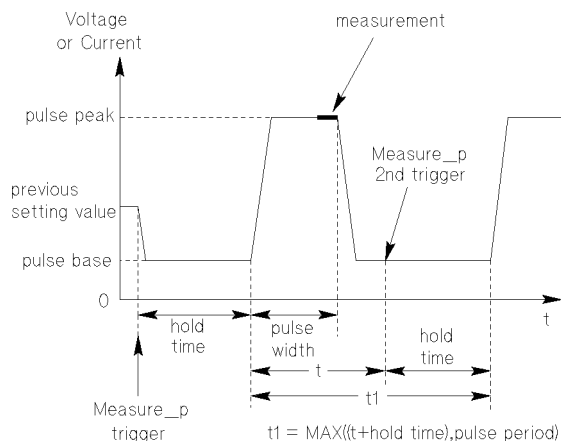
² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VS port.

³ *Width* must be less than 50% of *period*.

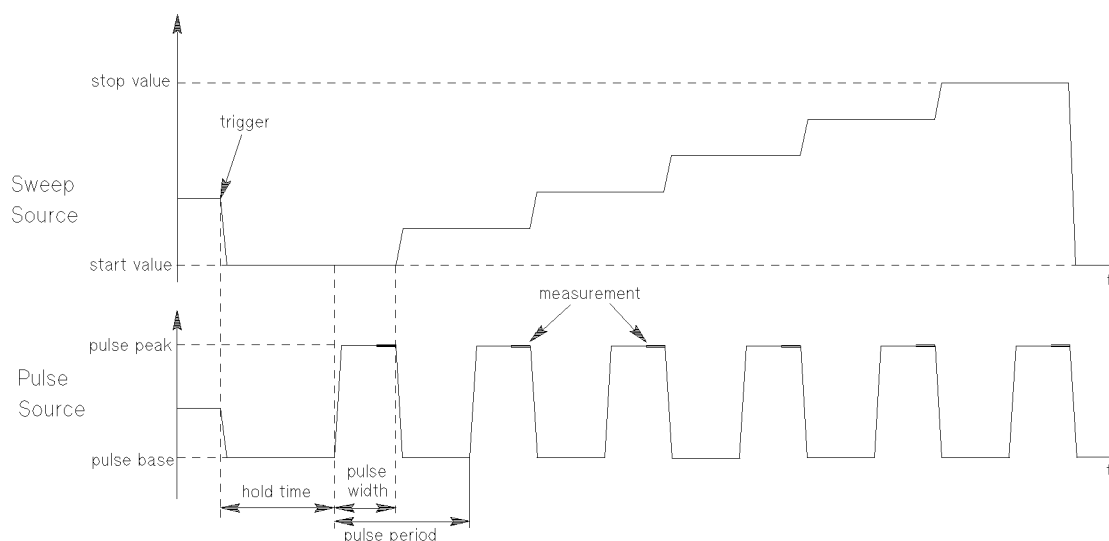
⁴ If *period* is set to “0”, a pulse period is not set.

Description

Base, *peak*, *width*, *period*, and *hold* determine the pulsed spot measurement and sweep measurement with pulsed bias conditions as shown in the following illustration:



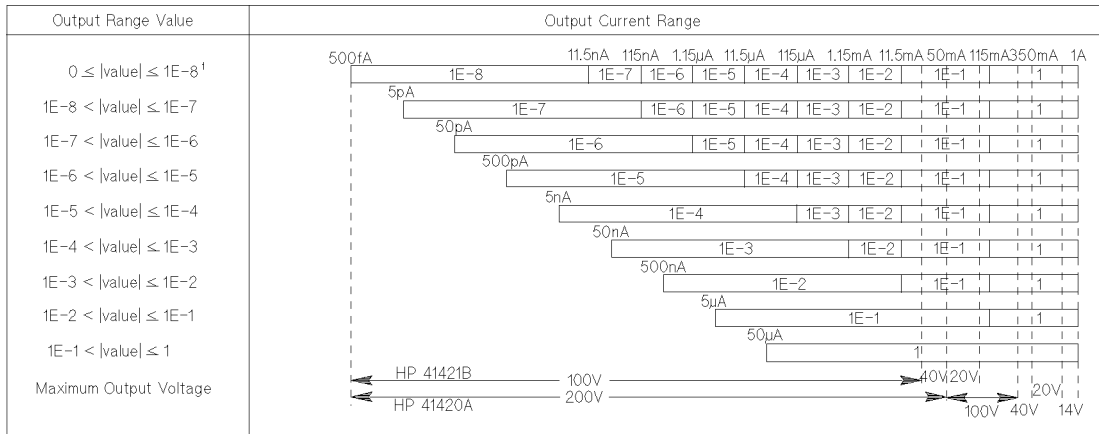
Pulsed Spot Measurements



Sweep with Pulsed Bias Measurements

The sweep source is synchronized with the pulsed bias and ignores the hold time and delay time settings of Set_iv. A pin number, unit address, or port address can be used to specify *port*. The current range is determined by the *range* value, as shown in the following table.

Set_pbias



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

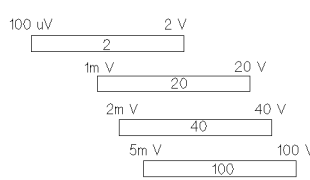
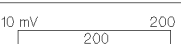
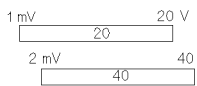
It is not necessary for the exact range to be defined for each unit. Specify an arbitrary value within the voltage range allowed and the system chooses the appropriate one.

If the specified voltage compliance is larger than 2 V, only current ranges greater than 100 μA can be set. You can use the REMAIN macro for *compliance*.

REMAIN specifies that the compliance remain at the present setting unless the unit is changed to the VS/IS mode by this function. If the specified unit changes VS/IS status, specifying the REMAIN macro sets the current compliance to 100 μA or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

If the specified current range can not output the *base* or *peak* value, the current range is set to the lowest range that can output the *base* or *peak* value.

The voltage range is determined by the *range* value, as shown in the following table. The maximum output voltage, voltage resolution, and current compliance range (maximum output current) depend on the set voltage range. (Current compliance can not be set for VS.)

Unit	Output Range Value	Output Voltage Range	Maximum Output Current	
HP 41420A HP 41421B	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 \leq value \leq 2$		HP 41420A	HP 41421B
	$2 < value \leq 20$		1 A	100 mA
	$20 < value \leq 40$		1 A (≤ 14 V) 7 00mA (> 14 V)	100 mA
	$40 < value \leq 100$		350 mA	50 mA
HP 41420A	$100 < value \leq 200$		50 mA	
VS (HP 41424A)	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 < value \leq 20$		100 mA	
	$20 < value \leq 40$		200 mA	

During a pulsed measurement, the DCS obtains a measurement result after each A/D conversion, and SMU filters are set to OFF, ignoring the settings of the Set_smu.

For pulsed spot measurements, execute a Measure_p function after you execute a Set_pbias function.

For sweep measurements with pulsed bias, execute a Sweep_iv function after you execute the Set_pbias and Set_iv functions. The sweep source must be set to the linear sweep mode, and you cannot set *power_compliance*.

If the following functions are executed for the specified unit after Set_pbias executes, the Set_pbias settings are lost. Accordingly, these functions must be executed to perform other sweep measurements after you perform a sweep measurement with pulsed bias.

Disable_port
Init_system
Set_iv

Set_pbias

Example

```
. . .
int err;
int number;
double vstart, vstop, hold_dmy, delay_dmy, comp_sweep;
double meas1[101], swp[101];
. . .
. . .
err = set_iv(24, LINEAR_V, 20.0, vstart, vstop,
             number, hold_dmy, delay_dmy,
             comp_sweep, 0.0, COMP_STOP);
err = set_pbias(28, V_SOURCE, 20.0, vbase, vpeak, width,
               period, hold, comp_bias);
err = sweep_iv(32, I_MEAS, 0.0, meas1, swp, NULL);
. . .
. . .
```

See Also

Measure_p

Set_smu

This function sets measurement integration time for SMUs, VSs, and VMs of the DCS.

Synopsis

```
#include <pcs/tis.h>

int set_smu(integ_time, filter, samples)
int integ_time, filter, samples;
```

Arguments

Item	Range Restrictions/Description
<i>integ_time</i>	Specifies the measurement integration time. Specify a definite number of integrations by combining this argument with the <i>samples</i> argument or choose from one of the following macros: INTEG_MANUAL (= 0) INTEG_SHORT (= 1) INTEG_MEDIUM (= 2) INTEG_LONG (= 3)
<i>filter</i>	Turns the SMU filter on or off. Use one of the following macros: FILTER_ON (= 1) FILTER_OFF (= 2)
<i>samples</i>	Specifies a number of samples to integrate the measurement value. 1 to 1023 ¹

¹ This argument is used only when *integ_time* is set to INTEG_MANUAL.

Description

To reduce measurement errors caused by line frequency noise or any other environmental noise source, the DCS takes a number of measurement samples and averages them to obtain measurement data. The number of measurement samples taken during each measurement is determined by the setting for the *integ_time* argument. Refer to the following table.

Set_smu

Integration Time	Description
INTEG_MANUAL	Measurement results are the average of the number of samples specified by the <i>samples</i> argument.
INTEG_SHORT	Measurement results are the average of the number of samples required to satisfy the specification of DCS. The number of samples depends on the voltage and current range. The maximum number of samples is 25.
INTEG_MEDIUM	Measurement results are the average of 32 samples taken during 1 line frequency period.
INTEG_LONG	Measurement results are the average of 512 samples taken during 16 line frequency periods.

The initial integration time setting is INTEG_SHORT.

If the *filter* argument of the SMU is set to FILTER_ON, the filter reduces overshoot voltage or current in the output of the SMU. If *filter* is set to FILTER_OFF, the SMU may generate an overshoot voltage or current, but higher speed measurements can be performed.

Example

```
. . .  
int err;  
. . .  
. . .  
err = set_smu(INTEG_LONG, FILTER_OFF, 0);  
. . .
```


Set_vm

This function sets the voltage measurement mode of the specified VM to either grounded measurement mode or differential measurement mode.

Synopsis

```
#include <pcs/tis.h>

int set_vm(port, mode)
int port, mode;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	An SWM port number, a unit address, or a pin number.
	<i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹
	<i>unit address</i> 32634, 32636, 32638, . . . 32664
	<i>pin number</i> 1 to 96 ²
<i>mode</i>	Choose from the following two macros for the measurement mode:
	VM_GND (= 0)
	VM_DIFF (= 1)

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Both VMs in the HP 41424A must be connected to the SWM, and one of the VMs must be connected to the specified pin.

Description

For VMs (HP 41424As), you can set either grounded voltage measurement mode or differential voltage measurement mode.

For a grounded measurement, the voltage between the VM terminal and circuit common is measured. Use zero (0) to reset all configured VMs to grounded mode. The SWM port must be connected to a VM. VM_GND *mode* sets the measurement to grounded mode.

For a differential measurement, the voltage between the VM1 and VM2 terminals of the HP 41424A is measured. For a differential measurement, specify the port address, unit address, or pin number of either VM1 or VM2 for the *port* argument. VM_DIFF *mode* sets the measurement unit to differential mode.

The *mode* argument is ignored if zero is specified for the *port* argument.

The voltage measurement ranges are determined by the Measure_v function. If you do not use the Measure_v function to set the voltage measurement ranges, the default is 20 V for grounded measurements and 2 V for differential measurements.

Set_vm

Example

```
. . .  
int err  
. . .  
err = set_vm(fnaux(1), VM_DIFF);  
. . .  
. . .
```

Short_cal

This function sets the SMUs, VSs, VMs, and AFU to their initial statuses. It also initiates the automatic calibration function for DCS, which measures and compensates for the dc offset voltages of the SMUs, VSs, VMs, A/D converters, and D/A converters.

Synopsis

```
#include <pcs/tis.h>

int short_cal()
```

Description

This function is identical to the Long_cal function. The SMUs and VSs of the DCS are set to their initial states, and calibration takes approximately 21 seconds for a standard system configuration.

You should perform Short_cal after your system has fully warmed up to correct for offsets caused by temperature drifts.

For best measurement results, perform Short_cal every 30 minutes or when the ambient temperature changes more than 3°C (6°F).

The system automatically performs calibration at about 30 minute intervals if all DCS units (HP 4142Bs) are in a disabled state for longer than 30 minutes.

Example

```
. . .
err=short_cal();
. . .
```

See Also

Init_system
Long_cal

Sweep Measurements

Introduction

The C Library provides the following functions for dc current and voltage sweep measurements:

- RswEEP_iv
- RswEEP_miv
- RswEEP_stop
- Set_iv
- Set_piv
- Set_sync
- Sweep_iv
- Sweep_miv

These functions set the parameters for I-V sweep measurements and perform the measurements according to the conditions. The functions are listed alphabetically in this chapter.

Note

When performing sweep measurements, specify the desired Set function before its corresponding Sweep function. For example, specify Set_iv before Sweep_iv is executed.

Rsweep_iv

This function performs real-time staircase sweep, synchronous staircase sweep, pulsed sweep, and staircase sweep with pulsed bias measurements.

Synopsis

```
#include <pcs/tis.h>

int rsweep_iv(port, mode, range, measure, status, source, sync)
int port, mode, *status;
double range, *measure, *source, *sync;
```

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, pin number, or unit address. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies whether a voltage measurement or a current measurement is performed on the <i>port</i> . Choose from the following macros: V_MEAS (= 1) I_MEAS (= 2)
<i>range</i> ³	Specifies a measurement range. 0.0 for Auto range mode OR: Numeric expression [A]: -1 to 1 ⁴
<i>measure</i>	A pointer to a variable of type double where the measurement value of the corresponding point is stored. Any valid pointer
<i>status</i>	A pointer to an integer where the measurement status code of the corresponding point is stored. This argument indicates one of the following: NORMAL_MEAS (= 0) DCS_COMP_OTHER (= 1) DCS_COMP (= 2) DCS_OSC (= 3) DCS_OVERFLOW (= 4) DCS_SWP_STOPPED(= 5)
<i>source</i>	A pointer to a double where the actual sweep source value of the corresponding point is stored. Specify NULL to discard the sweep source value (<i>sync</i> must also be NULL). Any valid pointer
<i>sync</i>	A pointer to a double where the secondary sweep source value of the corresponding point is stored.

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ If the measurement mode is V_MEAS (for voltage measurements), this is a dummy parameter because the V measurement range is defined by the set voltage compliance.

⁴ Actual range restrictions depend on the SMU or SMU port used.

Rswweep_iv

For VMs:

Item	Range Restrictions/Description
<i>port</i>	Specifies a port address, pin number, or unit address. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32634, 32636, 32638 . . . 32664 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies whether a voltage measurement or a current measurement is performed on the <i>port</i> . You can use the following macro: V_MEAS (= 1)
<i>range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: Numeric expression [V] Grounded measurements –40 to 40 Differential measurements –2 to 2
<i>measure</i>	A pointer to a variable of type double where the measurement value of the corresponding point is stored. Any valid pointer
<i>status</i>	A pointer to an integer where the measurement status code of the corresponding point is stored. This argument indicates one of the following: NORMAL_MEAS (= 0) DCS_COMP_OTHER (= 1) DCS_COMP (= 2) DCS_OSC (= 3) DCS_OVERFLOW (= 4) DCS_SWP_STOPPED (= 5)
<i>source</i>	A pointer to a double where the actual sweep source value of the corresponding point is stored. Specify NULL to discard the sweep source value (<i>sync</i> must also be NULL). Any valid pointer
<i>sync</i>	A pointer to a double where the secondary sweep source value of the corresponding point is stored. Any valid pointer.

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VM port.

Description

Rsweep_iv performs staircase sweep, synchronous staircase sweep, pulsed sweep, and staircase sweep with pulsed bias measurements according to the measurement parameters established in the following functions:

Measurement	Function Name
Staircase sweep measurements	Set_iv
Synchronous staircase sweep measurements	Set_iv, Set_sync
Pulsed sweep measurements	Set_piv
Staircase sweep with pulsed bias measurements	Set_iv, Set_pbias

Rsweep_iv returns measurement data as each sweep measurement finishes. Use the Rsweep_stop function to abort a sweep measurement.

A port address, unit address, or pin number can be used to specify the measurement *port* for Rsweep_iv.

An exact *range* value is not necessary. Specify an arbitrary value within the current range allowed and the system chooses the appropriate one.

Rsweep_iv obtains DCS data during a measurement and returns the measurement and status values. Status values are returned as follows:

Status Value	Condition
NORMAL_MEAS	Normal measurement end
DCS_COMP_OTHER	Another unit reached compliance
DCS_COMP	This unit reached compliance
DCS_OSC	This unit is oscillating
DCS_OVERFLOW	DC source measurement overflow
DCS_SWP_STOPPED	Sweep aborted by compliance condition

If the *stop_mode* = COMP_STOP of the Set_iv or Set_piv function, Rsweep_iv measurements are aborted when the measurement SMU(s) reach compliance. If this occurs before a measurement loop ends, Rsweep_iv returns 9999999.99999.

For Synchronous Sweep measurements only:

Rsweep_iv also obtains and returns primary (sweep port) and secondary (synchronous sweep port) values during a synchronous staircase sweep measurement. The DCS returns primary sweep values; secondary sweep values are calculated by Rsweep_iv and are stored in the *sync* array. The secondary sweep value calculation method is the same as that performed by the DCS, as follows:

Rswweep_iv

V sweep:

sweep mode			range of n	equation
linear sweep		single sweep	1 to N	$V_n = V_{start} + \frac{V_{stop} - V_{start}}{N - 1} (n - 1)$
		double sweep	N + 1 to 2N	$V_n = V_{stop} - \frac{V_{stop} - V_{start}}{N - 1} (n - N - 1)$
logarithmic sweep		single sweep	1 to N	$V_n = V_{start} \times EXP \left(\frac{\log_e \left(\frac{V_{stop}}{V_{start}} \right)}{N - 1} (n - 1) \right)$
		double sweep	N + 1 to 2N	$V_n = V_{stop} \times EXP \left(\frac{\log_e \left(\frac{V_{start}}{V_{stop}} \right)}{N - 1} (n - N - 1) \right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

I sweep:

sweep mode			range of n	equation
linear sweep		single sweep	1 to N	$I_n = I_{start} + \frac{I_{stop} - I_{start}}{N - 1} (n - 1)$
		double sweep	N + 1 to 2N	$I_n = I_{stop} - \frac{I_{stop} - I_{start}}{N - 1} (n - N - 1)$
logarithmic sweep		single sweep	1 to N	$I_n = I_{start} \times EXP \left(\frac{\log_e \left(\frac{I_{stop}}{I_{start}} \right)}{N - 1} (n - 1) \right)$
		double sweep	N + 1 to 2N	$I_n = I_{stop} \times EXP \left(\frac{\log_e \left(\frac{I_{start}}{I_{stop}} \right)}{N - 1} (n - N - 1) \right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

The *sync* argument should be NULL if *source* is NULL and the synchronous port is not defined by Set_sync, or if you want to discard the synchronous sweep source value.

In Offline Mode:

In offline mode, this function returns simulated measurement values to the *measure* variable, to the *status* variable, and to the *source* and *sync* variables. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns source, status, and calculated measurement values. Returned source values are a sequence of linear or logarithmic values in accordance with the *start* and *stop* values specified in the Set_iv or Set_piv function. Status value is always set to 0. Note that power compliance is not applicable in the offline mode. If *power_compliance* is specified in the Set_iv or Set_piv function, it is ignored.

Calculated measurement values are determined by the following equation:

$$nth \text{ measurement value} = \frac{\text{measurement value}}{\text{number of steps} - 1} \times (n - 1)$$

6-6 Sweep Measurements

Where: *number of steps* is set by the Set_iv or Set_piv function, and *measurement value* is determined as follows:

For current measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	= 0	I compliance	Same as SMU output value
	≠ 0	Range value	
VS	any	20 mA	Same as VS output value

For voltage measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	≠ 0	Range value	Same as measurement <i>range</i>

File Data Simulation mode:

This function reads <meas>, <status>, <source>, and <sync> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Rsweep_iv
[
<meas> <status> <source> <sync>
<meas> <status> <source> <sync>
. . . . .
<meas> <status> <source> <sync>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . .
int err;
. . .
for (i=0; i<number; i++) {
    err = rsweep_iv(12, I_MEAS, range, meas, stat,
                    sweep, NULL);
. . .
}
```

RswEEP_iv

See Also

RswEEP_miv

RswEEP_stop

Set_iv

Set_sync (for sweep measurements)

Sweep_iv

Sweep_miv

Rsweep_miv

This function performs real-time multichannel staircase sweep, synchronous staircase sweep, pulsed sweep, and staircase sweep with pulsed bias measurements. Multichannel synchronous staircase sweep measurements are described later in this section.

Synopsis

```
#include <pcs/tis.h>

int rsweep_miv(n, ports, ranges, measures, statuses, source, sync)
int n, ports[n], statuses[n];
double ranges[n], measures[n], *source, *sync;
```

Arguments

Item	Range Restrictions/Description
<i>n</i>	Specifies a number of ports where the sweep measurement is performed. This value determines the size of other array arguments, that is, <i>ports</i> , <i>ranges</i> , <i>statuses</i> , and <i>measures</i> . 1 through 8
<i>ports</i>	Specifies a pointer to an integer array that contains a list of SWM port addresses, pin numbers, or unit addresses. The array must have at least <i>n</i> elements.
<i>ranges</i>	Specifies a pointer to a double array, in which the elements contain a measurement range for the corresponding port. 0.0 for Auto range mode. The array must have at least <i>n</i> elements.
<i>measures</i>	A pointer to a double array where the measurement values of each measurement port are stored. The array must have at least <i>n</i> elements.
<i>statuses</i>	A pointer to an integer array where the measurement status code on each measurement port is stored. This argument indicates one of following: NORMAL_MEAS (= 0) DCS_COMP_OTHER (= 1) DCS_COMP (= 2) DCS_OSC (= 3) DCS_OVERFLOW (= 4) DCS_SWP_STOPPED(= 5)

Rswweep_miv

Item	Range Restrictions/Description
<i>source</i>	A pointer to a double where the actual sweep source value of the corresponding point is stored. Any valid pointer OR: Specify NULL to discard the sweep source value. (<i>sync</i> must also be NULL).
<i>sync</i>	A pointer to a double where the secondary sweep source value of the corresponding point is stored. Any valid pointer

Description

This function is for real-time multiple port sweep measurements. The function allows up to eight simultaneous real-time sweep measurements at once. The *n* argument specifies the number of ports to measure. The valid range of *n* is 1 through 8; if only one port is required, use the Rswweep_iv function.

Rswweep_miv performs multichannel staircase sweep measurements with the parameters established in the Set_iv function.

Rswweep_miv returns measurement data as each sweep measurement completes. Use the Rswweep_stop function to abort a sweep measurement.

A port address, unit address, or pin number can be used to specify the measurement ports of Rswweep_miv. Measurement and sweep source ports must be already connected to the specified measurement units. Refer to the Measure_i and Measure_v functions for details on measurement range.

Rswweep_miv obtains DCS data during a measurement and returns measurement and status values to their corresponding arrays. Status values are returned as follows:

Status Value	Condition
NORMAL_MEAS	Normal measurement end
DCS_COMP_OTHER	Another unit reached compliance
DCS_COMP	This unit reached compliance
DCS_OSC	This unit is oscillating
DCS_OVERFLOW	DC source measurement overflow
DCS_SWP_STOPPED	Sweep is stopped by the compliance condition

If the *stop_mode* = COMP_STOP of the Set_iv or Set_piv function, Rswweep_miv measurements are aborted when the measurement SMU(s) reach compliance.

The number of measurement ports is specified by the *n* argument. The number of elements in the *ports*, *ranges*, *measures*, and *statuses* arrays must be equal to or greater than *n*.

Exact *ranges* value is not necessary. Specify an arbitrary value within the current range allowed and the system chooses the appropriate one.

6-10 Sweep Measurements

RswEEP_miv performs multichannel synchronous staircase sweep measurements according to the measurement parameters set in the Set_iv and Set_sync functions.

RswEEP_miv also obtains and returns primary and secondary values during a synchronous staircase sweep measurement. The DCS returns primary sweep values, and secondary sweep values are calculated by RswEEP_miv and are stored in the *sync*. Secondary value calculation method is the same as that performed by the DCS, as follows:

V sweep:

sweep mode		range of n	equation
linear sweep	single sweep	1 to N	$V_n = V_{start} + \frac{V_{stop} - V_{start}}{N - 1} (n - 1)$
	double sweep	N + 1 to 2N	$V_n = V_{stop} - \frac{V_{stop} - V_{start}}{N - 1} (n - N - 1)$
logarithmic sweep	single sweep	1 to N	$V_n = V_{start} \times EXP\left(\frac{\log_e\left(\frac{V_{stop}}{V_{start}}\right)}{N - 1} (n - 1)\right)$
	double sweep	N + 1 to 2N	$V_n = V_{stop} \times EXP\left(\frac{\log_e\left(\frac{V_{start}}{V_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

I sweep:

sweep mode		range of n	equation
linear sweep	single sweep	1 to N	$I_n = I_{start} + \frac{I_{stop} - I_{start}}{N - 1} (n - 1)$
	double sweep	N + 1 to 2N	$I_n = I_{stop} - \frac{I_{stop} - I_{start}}{N - 1} (n - N - 1)$
logarithmic sweep	single sweep	1 to N	$I_n = I_{start} \times EXP\left(\frac{\log_e\left(\frac{I_{stop}}{I_{start}}\right)}{N - 1} (n - 1)\right)$
	double sweep	N + 1 to 2N	$I_n = I_{stop} \times EXP\left(\frac{\log_e\left(\frac{I_{start}}{I_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

In Offline Mode:

In offline mode, this function returns simulated measurement values to the *measures* array, to the *statuses* array, and to *source* and *sync*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns source, status, and calculated measurement values. Returned source values are a sequence of linear or logarithmic values in accordance with the *start* and *stop* values specified in the Set_iv or Set_piv function. Status value is always set to 0. Note that power compliance is not applicable in the offline mode: If *power_compliance* is specified in the Set_iv or Set_piv function, it is ignored.

Rswweep_miv

Calculated measurement values are determined by the following equation:

$$nth \text{ measurement value} = \frac{\text{measurement value}}{\text{number of steps} - 1} \times (n - 1)$$

Where *number* of steps is set by the Set_iv or Set_piv function, and measurement value is determined as follows:

For current measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	= 0	I compliance	Same as SMU output value
	≠ 0	Range value	
VS	any	20 mA	Same as VS output value

For voltage measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	≠ 0	Range value	Same as measurement <i>range</i>

File Data Simulation mode:

This function reads <meas>, <status>, <source>, and <sync> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Rswweep_miv
[
<meas1> <meas2> . . . <st1> <st2> . . . <source> <sync>
<meas1> <meas2> . . . <st1> <st2> . . . <source> <sync>
. . . . .
<meas1> <meas2> . . . <st1> <st2> . . . <source> <sync>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . .
. . .
err = set_iv(8, LINEAR_V, 20, vstart, vstop, number,
            hold, delay, comp, 0.0, COMP_CONT);
for (i=0; I<number; i++) {
    err = rsweep_miv(n, ports, ranges,
                    &measures, &stat, &sweep, NULL);
. . .
}
```


See Also

RswEEP_iv
RswEEP_stop
Set_iv
Set_sync (for sweep measurements)
Sweep_iv
Sweep_miv

Rsweep_stop

This function stops real time sweep measurements.

Synopsis

```
#include <pcs/tis.h>

int rsweep_stop()
```

Description

This function is used in conjunction with the Rsweep_iv and Rsweep_miv functions to stop real time sweep measurements. After this function executes, the sweep start value is set to the SMU(s) or VS(s), but sweep conditions are not changed at all and the output relay of the measurement unit is not disconnected.

Example

```
. . .
err = set_iv(8, LINEAR_V, 20, vstart, vstop, number, hold,
            delay, comp, 0.0, COMP_CONT);
for (i=0; I<number; i++) {
    err = rsweep_miv(n, ports, ranges, &measures, &stat,
                    &sweep, NULL);
    if (err == -1) {
        rsweep_stop();
        break;
    }
}
. . .
. . .
```

See Also

Rsweep_iv
Rsweep_miv

Set_iv

This function sets the sweep parameters for I-V measurements.

Synopsis

```
#include <pcs/tis.h>

int set_iv(port, sweep_mode, range, start, stop, number,
           hold, delay, compliance, power_compliance, stop_mode)
int port, sweep_mode, number, stop_mode;
double range, start, stop, hold, delay;
double compliance, power_compliance;
```

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or a unit address corresponding to an SMU or VS. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>sweep_mode</i>	Determines the voltage source or current source mode of the SMU, the linear or logarithmic sweep mode, and single or double (bi-directional) staircase mode. Choose from the following macros: LINEAR_V (= 1) LINEAR_I (= 2) LOG_V (= -1) LOG_I (= -2) LINEAR_V_DBL (= 3) LINEAR_I_DBL (= 4) LOG_V_DBL (= -3) LOG_I_DBL (= -4)
<i>range</i>	Voltage or current force range. 0.0 for Auto range mode OR: For voltage source: numeric expression [V] -200 to 200 ³ For current source: numeric expression [A] -1 to 1 ³
<i>start, stop</i>	Sweep start and stop voltage or current value. These values should have the same polarity for logarithmic sweep. Numeric expression [V]: -200 to 200 ³
<i>number</i>	Number of sweep steps. 2 to 1001

Set_iv

Item	Range Restrictions/Description
<i>hold</i>	<p>Specifies a wait time is taken prior to the trigger of the measurement of the first step after the force value is set to <i>start</i>.</p> <p>If <i>delay</i> is also a non-zero, the additional <i>delay</i> time follows after the first step.</p> <p>Numeric expression [s]: 0 to 65.535 Resolution: 0.01</p>
<i>delay</i>	<p>If this value is greater than 0.0, a wait time is taken at each measurement step after the force value changes.</p> <p>Numeric expression [s]: 0 to 65.535 Resolution: 0.001</p>
<i>compliance</i>	<p>Specifies a voltage or current compliance value according to the <i>sweep_mode</i>.</p> <p>REMAIN macro OR: Numeric expression [A]: -1 to $1$³</p>
<i>power_compliance</i>	<p>Limits the power (production of voltage and current being force or measured) applied to the <i>port</i>.</p> <p>Specify 0.0 to allow power up to the SMU maximum rating OR: Numeric expression [W]: 0.001 to $14$⁴ Resolution: 0.001</p>
<i>stop_mode</i>	<p>Specifies whether the sweep measurement should be aborted upon a compliance condition, or continue measurement until the last step specified by <i>number</i>.</p> <p>Ignored when non-zero <i>power_compliance</i> is specified. In this case, <i>stop_mode</i> defaults to the COMP_STOP mode.</p> <p>Choose from one of the following macros:</p> <p>COMP_CONT (= 1) COMP_STOP (= 2)⁵</p>

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ Actual *range*, *start*, *stop*, and *compliance* range restrictions depend on the SMU and SMU port used. *Compliance* depends on the set voltage range.

⁴ If you set *stop_mode* to COMP_CONT, set *power_compliance* to 0.

⁵ You can set *power_compliance* up to 14 W for the HP 41420A SMU connected to the SMU2 port only. For other SMUs, you can set *power_compliance* up to 2 W.

For VS voltage sweep measurements:

Item	Range Restrictions/Description
<i>port</i>	An SWM port address, a pin number, or a unit address corresponding to an SMU or VS. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32633, 32635, 32537 . . . 32663 <i>pin number</i> 1 to 96 ²
<i>sweep_mode</i>	Determines the voltage source or current source mode of the SMU, the linear or logarithmic sweep mode, and single or double (bi-directional) staircase mode. Choose from the following macros: LINEAR_V (= 1) LINEAR_I (= 2) LOG_V (= -1) LOG_I (= -2) LINEAR_V_DBL (= 3) LINEAR_I_DBL (= 4) LOG_V_DBL (= -3) LOG_I_DBL (= -4)
<i>range</i>	Voltage force range. 0.0 for Auto range mode OR: Numeric expression: -40 to 40
<i>start, stop</i>	Specify sweep start and stop voltage or current value. These values should have the same polarity for logarithmic sweep. Numeric expression [V]: -40 to 40
<i>number</i>	Number of sweep steps. 2 to 1001
<i>hold</i>	Specifies a wait time is taken prior to the trigger of the measurement of the first step after the force value is set to <i>start</i> . If <i>delay</i> is a non-zero, the additional <i>delay</i> time follows after the first step. Numeric expression [s]: 0 to 65.535 Resolution: 0.01

Set_iv

Item	Range Restrictions/Description
<i>delay</i>	<p>If this value is greater than 0.0, a wait time is taken at each measurement step after the force value changes.</p> <p>Numeric expression [s]: 0 to 65.535 Resolution: 0.001</p>
<i>compliance</i>	<p>Specifies a voltage or current compliance value according to the <i>sweep_mode</i>.</p> <p>REMAIN macro OR: ³ Numeric expression</p>
<i>power_compliance</i>	<p>Limits the power (production of voltage and current being force or measured) applied to the <i>port</i>.</p> <p>Numeric expression—³</p>
<i>stop_mode</i>	<p>Specifies whether the sweep measurement should be aborted upon a compliance condition, or continue measurement until the last step specified by <i>number</i>.</p> <p>Ignored when non-zero <i>power_compliance</i> is specified. In this case, <i>stop_mode</i> defaults to the COMP_STOP mode.</p> <p>Choose from one of the following macros:</p> <p>COMP_CONT (= 1) COMP_STOP (= 2)</p>

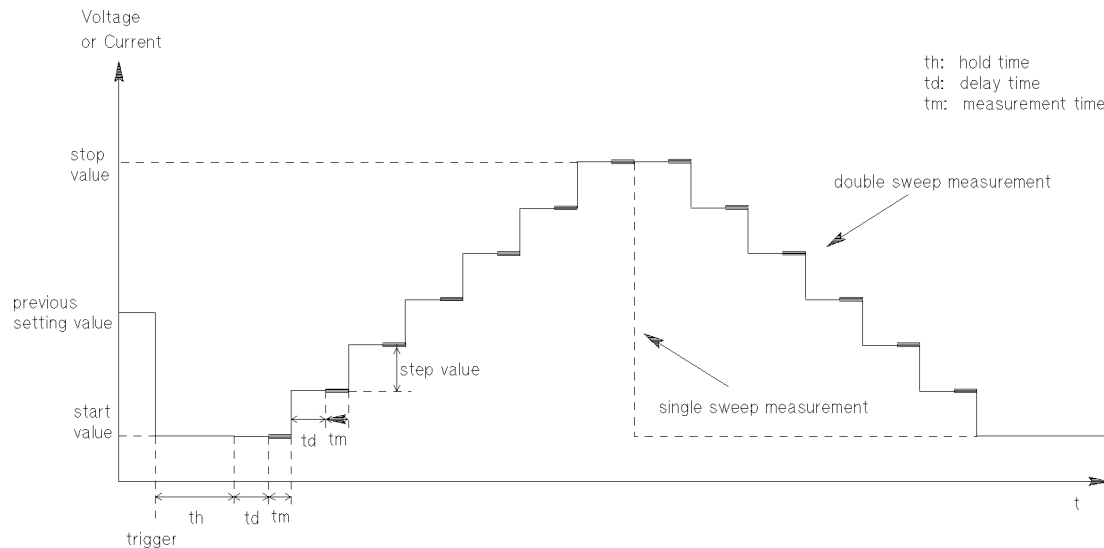
¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VS port.

³ These are dummy parameters.

Description

When Sweep_iv executes, *start* (voltage or current), *stop*, *hold*, and *delay* determine the sweep measurement conditions, as shown in the following illustration.



The start, stop, and step values are determined by the rules in the following table.

In the linear sweep mode (*sweep_mode*)

Item	Description
start value	resolution: depends on the set voltage or current range.
stop value ¹	$\text{stop value} = \text{start value} + \text{step value} \times (\text{number of steps} - 1)$
step value	$\text{step value} = \frac{\text{stop value} - \text{start value}}{\text{number of steps} - 1}$ resolution: depends on the set voltage or current range. maximum: $\text{stop} - \text{start}$

¹ The actual stop value may be slightly different than the specified value. Refer to the Set_cv explanation for more details.

Set_iv

In the log sweep mode (*sweep_mode*)

Item	Description
start value	resolution: depends on the set voltage or current range.
stop value ¹	$\text{stop value} = \text{start value} \times 10^{\frac{\text{step ratio} \times (\text{number of steps} - 1)}{20}}$
step ratio	$\text{step ratio} = \frac{20 \times \log_{10} \frac{\text{stop value}}{\text{start value}}}{\text{number of steps} - 1}$ resolution: 0.02 dB/decade maximum: 20 dB/decade

¹ The actual stop value may be slightly different than the specified value. Refer to the Set_cv explanation for more details.

A pin number, unit address, or a port address can be used to specify the *port*.

In the *sweep_mode*, a linear sweep or a logarithmic sweep is made depending on which of the following macros you specify for *sweep_mode*:

LINEAR_V:	Performs linear voltage sweep.
LINEAR_I:	Performs linear current sweep.
LOG_V:	Performs logarithmic voltage sweep.
LOG_I:	Performs logarithmic current sweep.
LINEAR_V_DBL:	Performs linear voltage double sweep.
LINEAR_I_DBL:	Performs linear current double sweep.
LOG_V_DBL:	Performs logarithmic voltage double sweep.
LOG_I_DBL:	Performs logarithmic current double sweep.

The output of the current range is determined by the *range* value, as shown in the following table.

Output Range Value	Output Current Range
$0 \leq \text{value} \leq 1\text{E-}9^1$	50fA 1E-9 1.15nA 1E-8 115nA 1E-7 1.15μA 1E-6 115μA 1E-5 1.15mA 1E-4 115mA 1E-3 1.15A 1E-2 115A 1E-1 1A
$1\text{E-}9 < \text{value} \leq 1\text{E-}8^1$	500fA 1E-8 1.15nA 1E-7 115nA 1E-6 1.15μA 1E-5 1.15mA 1E-4 115mA 1E-3 1.15A 1E-2 115A 1E-1 1A
$1\text{E-}8 < \text{value} \leq 1\text{E-}7$	5pA 1E-7 1.15nA 1E-6 115nA 1E-5 1.15μA 1E-4 115μA 1E-3 1.15mA 1E-2 115mA 1E-1 1A
$1\text{E-}7 < \text{value} \leq 1\text{E-}6$	50pA 1E-6 1.15nA 1E-5 115nA 1E-4 1.15μA 1E-3 1.15mA 1E-2 115mA 1E-1 1A
$1\text{E-}6 < \text{value} \leq 1\text{E-}5$	500pA 1E-5 1.15nA 1E-4 115nA 1E-3 1.15μA 1E-2 1.15mA 1E-1 115mA 1A
$1\text{E-}5 < \text{value} \leq 1\text{E-}4$	5nA 1E-4 1.15nA 1E-3 1.15μA 1E-2 1.15mA 1E-1 115mA 1A
$1\text{E-}4 < \text{value} \leq 1\text{E-}3$	50nA 1E-3 1.15nA 1E-2 1.15μA 1E-1 115mA 1A
$1\text{E-}3 < \text{value} \leq 1\text{E-}2$	500nA 1E-2 1.15nA 1E-1 115mA 1A
$1\text{E-}2 < \text{value} \leq 1\text{E-}1$	5μA 1E-1 115mA 1A
$1\text{E-}1 < \text{value} \leq 1$	50μA 1 115mA 1A
Maximum Output Voltage	HP 41421B 100V 40V 20V 10V 40V 14V HP 41420A 200V

¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

6-20 Sweep Measurements

When the *sweep_mode* is linear, the current range during the sweep is set depending on the value of the *start* or *stop*, whichever is larger.

An exact *range* value is not necessary. Specify an arbitrary value within the current range allowed and the system chooses the appropriate one.

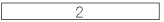
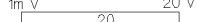





If you set the *range* to 0 when the *sweep_mode* is logarithmic, the optimum current range is selected depending on the current value output during the sweep. If you set the *range* to a value other than 0, however, the current range covers a range lower than the specified range. Therefore, if the sweep covers more than four decades, set the *range* to 0.

The voltage range of an SMU set to the IS mode is determined by the voltage *compliance*. Also, a pulse that may reach voltage compliance can be output during the current range changing of an SMU, so do not set the voltage *compliance* to an unnecessarily large value.

You can use the REMAIN macro for *compliance*. REMAIN specifies that the compliance remain at the present setting unless the unit changes voltage or current source mode by this function. If the specified unit changes voltage or current source status, specifying the REMAIN macro sets the current compliance to 100 μ A or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

The voltage range is set depending on the value of *range*. The maximum output voltage, voltage resolution, and current compliance range (maximum output current) depends on the set voltage range. (Current compliance cannot be set for VS.)

The ranges of start value and stop value depend on the unit. See the following table.

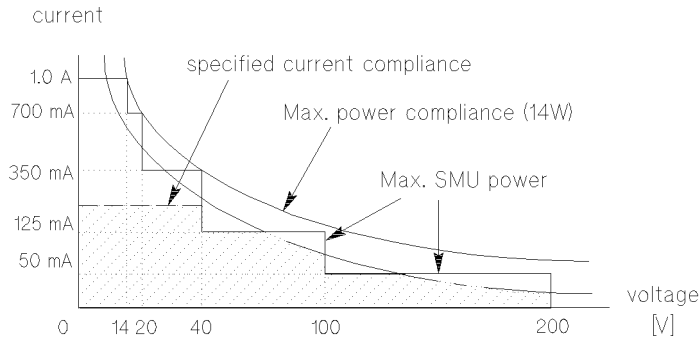
Unit	Output Range Value	Output Voltage Range	Maximum Output Current	
HP 41420A	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 \leq \text{value} \leq 2$	100 μ V  2 V	HP 41420A	HP 41421B
	$2 < \text{value} \leq 20$	1m V  20 V	1 A	100 mA
	$20 < \text{value} \leq 40$	2m V  40 V	1 A (≤ 14 V) 7 00mA (> 14 V)	100 mA
HP 41421B	$40 < \text{value} \leq 100$	5m V  100 V	350 mA	50 mA
			125 mA	20 mA
HP 41420A	$100 < \text{value} \leq 200$	10 mV  200 V	50 mA	
VS (HP 41424A)	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 < \text{value} \leq 20$	1 mV  20 V	100 mA	
	$20 < \text{value} \leq 40$	2 mV  40 V	200 mA	

The ranges of current and voltage compliance depend on the output voltage or current range. Refer to Description for Force_v or Force_i for details. If you set *power_compliance*, the compliance can be set up to the maximum value of the unit used, independent of the output voltage or current range.

Power compliance is the programmable output power limit of the SMU. Its function is to protect test devices against damage caused by excessive power. If *power_compliance* is set, an

Set_iv

SMU can output a voltage or current within the specified *power_compliance* as shown in the following illustration.



Stop_mode defines whether to continue or abort a voltage or current sweep if the output reaches the specified voltage or current compliance or current limit.

If you set *stop_mode* to COMP_CONT, voltage or current is swept to the last specified step, even if an SMU reaches voltage or current compliance.

If you set *stop_mode* to COMP_STOP, the voltage or current sweep aborts and returns dummy data (9999999.99999) if an SMU reaches voltage or current compliance.

If you set *power_compliance* to a non-zero value, the sweep aborts and returns dummy data independent of the specified *stop_mode* when an SMU reaches *power_compliance*, or *compliance*. *Stop_mode* defaults to the COMP_STOP mode.

Note



Sufficient data storage space must be passed to a subsequent Sweep_iv or Sweep_miv function. The data space must accommodate the *number* of double precision floating number for the single sweep mode and $2 \times \text{number}$ of double precision floating values for the double sweep mode.

Example

```
. . .
. . .
if (set_iv(drain, LINEAR_V, 20.0, 0.0, -10.0, 20, 0.1,
          0.05, 1e-2, 0.0, COMP_STOP)
    == -1) error_rep()
. . .
. . .
int error_rep()
. . .
. . .
```

See Also

Set_piv
Sweep_iv
Sweep_miv

Set_piv

This function sets the parameters for SMU and VS pulsed sweep measurements. To execute pulsed sweep measurements, use the Sweep_iv function.

Synopsis

```
#include <pcs/tis.h>

int set_piv(port, sweep_mode, range, base, start,
            stop, number, width, period, hold,
            compliance, stop_mode)
int port, sweep_mode, number, stop_mode;
double range, base, start, stop;
double width, period, hold, compliance;
```

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or a unit address.
	<i>port address</i> 32701 to 32704, 32729, 32732 ¹
	<i>unit address</i> 32621 to 32632
	<i>pin number</i> 1 to 96 ²
<i>sweep_mode</i>	Specifies either the VS or IS mode of the SMU, and either single or double stair mode.
	Logarithmic sweep is not allowed.
	Use one of the following macros:
	LINEAR_V (= 1)
	LINEAR_I (= 2)
	LINEAR_V_DBL (= 3)
	LINEAR_I_DBL (= 4)

Set_piv

Item	Range Restrictions/Description
<i>range</i>	Specifies either the voltage or current force range. 0.0 for Auto range mode OR: For voltage source: numeric expression [V] -200 to 200 ³ For current source: numeric expression [A] -1 to 1 ³
<i>base</i>	Specifies the pulse base voltage or current in which the <i>port</i> is reset to on each sweep step. For voltage source: numeric expression [V] -200 to 200 ³ For current source: numeric expression [A] -1 to 1 ³
<i>start, stop</i>	Specify the sweep start and stop values, which define the peak voltage/current of each pulse. For voltage source: numeric expression [V] -200 to 200 ³ For current source: numeric expression [A] -1 to 1 ³
<i>number</i>	Specifies the number of sweep steps. 2 to 1001
<i>width, period</i>	Specifies the pulse timing parameters. Numeric expression [s]: 0.001 to 0.05 ⁴ Resolution: 1E-4
<i>hold</i>	The dc source waits the number of seconds defined in this argument before starting the sweep measurement after it is triggered and the force value is set to <i>base</i> . This wait time applies only to the first step of the sweep. Numeric expression [s]: 0 to 655.35 Resolution: 0.01
<i>compliance</i>	Specifies the voltage or current compliance on the <i>port</i> . REMAIN macro OR: For voltage source: numeric expression [A] -1 to 1 ³ For current source: numeric expression [V] -200 to 200 ³
<i>stop_mode</i>	Specifies one of two conditions: either the pulsed sweep measurement is aborted upon reaching a compliance condition, or it continues the measurement until the last step specified by <i>number</i> . Choose one of the following macros: COMP_CONT (= 1) COMP_STOP (= 2)

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ Actual range restrictions of these values depend on the unit used. The values of *base*, *start*, *stop*, and *compliance* depend on the specified voltage range.

⁴ *Width* must be less than 50% of *period*.

For VSs:

Item	Range Restrictions/Description
<i>port</i>	Specifies a port address, a pin number, or a unit address. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32633, 32635, 32637 . . . 32663 <i>pin number</i> 1 to 96 ²
<i>sweep_mode</i>	Specifies either the VS or IS mode of the SMU, and either single or double stair mode. Logarithmic sweep is not allowed. Use one of the following macros: LINEAR_V (= 1) LINEAR_I (= 2) LINEAR_V_DBL (= 3) LINEAR_I_DBL (= 4)
<i>range</i>	Specifies either the voltage or current force range. 0.0 for Auto range mode OR: Numeric expression [V]: -40 to 40
<i>base</i>	Specifies the pulse base voltage or current in which the <i>port</i> is reset to on each sweep step. Numeric expression [V]: -40 to 40
<i>start, stop</i>	Specifies the sweep start and stop values, which define the peak voltage/current of each pulse. Numeric expression [V]: -40 to 40
<i>number</i>	Specifies the number of sweep steps. 2 to 1001

Set_piv

Item	Range Restrictions/Description
<i>width, period</i>	Specifies the pulse timing parameters. Numeric expression [s]: 0.001 to 0.05 ³ Resolution: 1E-4
<i>hold</i>	The dc source waits the number of seconds defined in this argument before starting the sweep measurement after it is triggered and the force value is set to <i>base</i> . This wait time applies only to the first step of the sweep. Numeric expression [s]: 0 to 655.35 Resolution: 0.01
<i>compliance</i>	Specifies the voltage or current compliance on the <i>port</i> . REMAIN macro OR: Numeric expression:— ⁴
<i>stop_mode</i>	Specifies one of two conditions: either the pulsed sweep measurement is aborted upon reaching a compliance condition, or it continues the measurement until the last step specified by <i>number</i> . Choose one of the following macros: COMP_CONT (= 1) COMP_STOP (= 2)

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

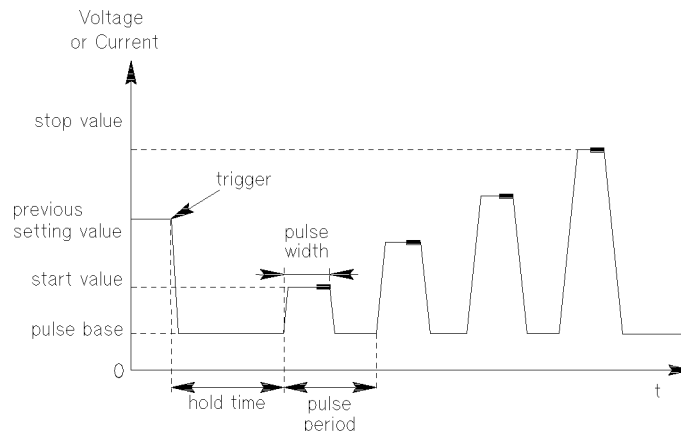
² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VS port.

³ *Width* must be less than 50% of *period*.

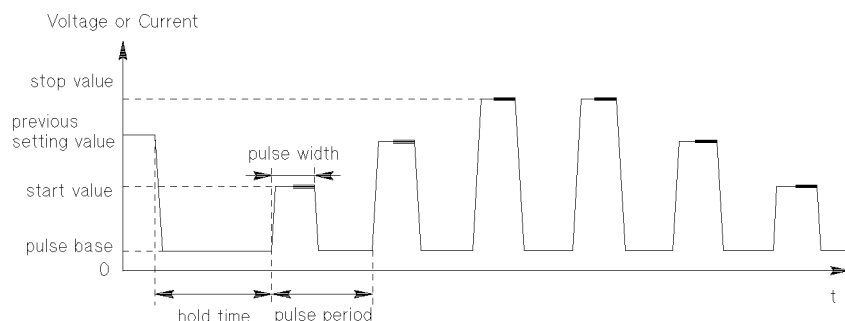
⁴ This is a dummy parameter.

Description

When Sweep_piv is executed, *base*, *start*, *stop*, *width*, *period*, and *hold* of the Set_piv function determine the pulsed sweep measurement conditions as shown in the following illustrations.



Single Pulsed Sweep (pulse sweep mode = 1 or 2)



Double Pulsed Sweep (pulse sweep mode = 3 or 4)

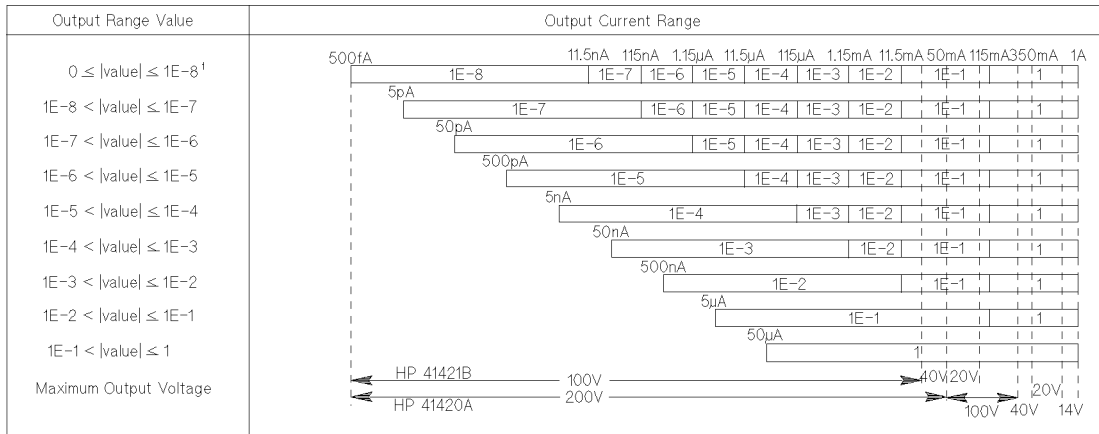
Item	Description
start value	resolution: depends on the set voltage or current range
stop value ¹	$\text{stop value} = \text{start value} + \text{step value} \times (\text{number of steps} - 1)$
step value	$\text{step value} = \frac{\text{stop value} - \text{start value}}{\text{number of steps} - 1}$

¹ The actual stop value may be slightly different than the specified value. Refer to the Set_cv explanation for more details.

A pin number, unit address, or port address can be used to specify *port*. A single pulsed sweep is selected when the *sweep_mode* is LINEAR_V or LINEAR_I. A double pulsed sweep is selected when the *sweep_mode* is LINEAR_V_DBL or LINEAR_I_DBL.

The current range is determined by the *range* value, as shown in the following table.

Set_piv



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

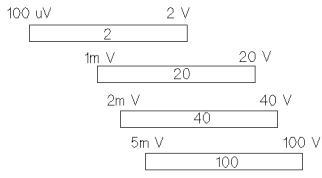
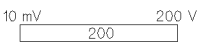
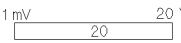

An exact *range* value is not necessary. Specify an arbitrary value within the current range allowed and the system chooses the appropriate one.

If the *sweep_mode* is LINEAR_I or LINEAR_I_DBL, the current range during a pulsed sweep is set depending on the value of the pulse base current, start current or stop current, whichever is largest. If the specified *compliance* is larger than 2 V, only current ranges greater than 100 μA can be set.

The voltage range of an SMU set to the IS mode is determined by the voltage *compliance*. Be aware that a pulse may be output during an current range changing of an SMU that may reach voltage compliance, so do not set voltage *compliance* to an unnecessarily large value.

If the *sweep_mode* is LINEAR_V or LINEAR_V_DBL, the voltage range is set depending on the value of the *range*. The maximum output voltage, voltage resolution, and current compliance range (maximum output current) depends on the set voltage range. (Current compliance cannot be set for VS.)

The ranges of *start* and *stop*, depending on the unit used, are shown in the following table. The ranges of current *compliance* and voltage *compliance* depend on the output voltage or current range.

Unit	Output Range Value	Output Voltage Range	Maximum Output Current	
HP 41420A HP 41421B	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 \leq value \leq 2$		HP 41420A	HP 41421B
	$2 < value \leq 20$		1 A 1 A (≤ 14 V) 700 mA (> 14 V)	100 mA
	$20 < value \leq 40$		350 mA	50 mA
	$40 < value \leq 100$		125 mA	20 mA
HP 41420A	$100 < value \leq 200$		50 mA	
VS (HP 41424A)	0	The optimum range is selected depending on the value of the start voltage or stop voltage, whichever is larger.		
	$0 < value \leq 20$		100 mA	
	$20 < value \leq 40$		200 mA	

Stop_mode defines whether to continue or abort a pulsed voltage or current sweep if the output reaches the specified voltage or current *compliance* or current limit.

If you set *stop_mode* to COMP_CONT, pulsed voltage or current is swept to the last specified step, even if an SMU reaches voltage or current *compliance*.

If you set *stop_mode* to COMP_STOP, the pulsed voltage or current sweep aborts and returns dummy data (9999999.99999) if an SMU reaches voltage or current *compliance*.

Example

```
. . .
int err;
int number;
double vbase, vstart, vstop, width, period, hold, comp;
double meas1[101], swp[101];
. . .
. . .
err = set_piv(24, LINEAR_V, 20.0, vbase, vstart, vstop,
             number, width, period, hold, comp, COMP_STOP);
err = sweep_iv(32, I_MEAS, 0.0, meas1, swp1, NULL);
. . .
. . .
```

See Also

Rsweep_iv
Set_iv
Set_pbias
Set_sync (for sweep measurements)
Sweep_iv

Set_sync (for sweep measurements)

This function sets the synchronous port for performing synchronous staircase sweep measurements.

Synopsis

```
#include <pcs/tis.h>

int set_sync(port, mode, offset, ratio, compliance,
             power_compliance)
int port, mode;
double offset, ratio, compliance, power_compliance;
```

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or a unit address of a synchronous sweep port. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies the polarity of the synchronization. Choose from one of the following macros: POS_SYNC (= 0) NEG_SYNC (= 1)
<i>offset</i>	Specifies the offset of the synchronous value relative to the primary value. For voltage synchronous source: numeric expression [V] -400 to 400 ³ For current synchronous source: numeric expression [A] -2 to 2 ³
<i>ratio</i>	Specifies the gradient ratio of the synchronous value, relative to the primary value. If a negative ratio is desired, use NEG_SYNC for the <i>mode</i> argument. Numeric expression: 0.01 to 10 Resolution: 0.01
<i>compliance</i>	Specifies the voltage or current compliance on the synchronous port. REMAIN macro OR: For voltage synchronous source: numeric expression [A] -1 to 1 ³ For current synchronous source: numeric expression [V] -200 to 200 ³
<i>power compliance</i>	Limits the power (production of voltage and current being force or measured) applied to the <i>port</i> . Specify 0.0 for power up to the SMU maximum rating OR: Numeric expression [W]: 0.001 to 14 ⁴ Resolution: 0.001

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ Actual *offset* value restriction depends on the SMU and SMU port used.

⁴ You can set *power_compliance* up to 14 W for an HP 41420A SMU that is connected to an SMU2 port only. For other SMUs, you can set *power_compliance* up to 2 W.

Set_sync (for sweep measurements)

For VS voltage sweep:

Item	Range Restrictions/Description
<i>port</i>	Specifies a port address, a pin number, or a unit address of a synchronous sweep port. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32633, 32635, 32637 . . . 32663 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies the polarity of the synchronization. Choose from one of the following macros: POS_SYNC (= 0) NEG_SYNC (=1)
<i>offset</i>	Specifies the offset of the synchronous value relative to the primary value. Numeric expression [V]: -80 to 80
<i>ratio</i>	Specifies the gradient ratio of the synchronous value, relative to the primary value. If a negative ratio is desired, use NEG_SYNC for the <i>mode</i> argument. Numeric expression: 0.01 to 10 Resolution: 0.01
<i>compliance</i>	Specifies the voltage or current compliance on the synchronous port. REMAIN macro OR: Numeric expression [A]:— ³
<i>power_compliance</i>	Limits the power (production of voltage and current being force or measured) applied to the <i>port</i> . Specify 0.0 for power up to the SMU maximum rating OR: Numeric expression [W]:— ⁴

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VS port.

³ This is a dummy parameter. If output range is 20 V, current is limited to 100 mA. If the output range is 40 V, current is limited to 20 mA.

⁴ This is a dummy parameter (cannot set *power_compliance* for VSs).

Description

This function used in conjunction with the Set_iv function sets the control parameters for the synchronous sweep port (SMU or VS).

The output mode of the synchronous sweep port is automatically set to the same output mode (VS or IS) and sweep type (linear or logarithmic) as the sweep port in the Set_iv function. If the *mode* of the Set_iv function is set to perform a logarithmic sweep, the *offset* setting of this function is ignored.

Choose from one of the following macros for the *mode*:

POS_SYNC: The synchronous value rises when the primary sweep rises and drops when the primary value drops.

NEG_SYNC: The synchronous value drops when the primary sweep source value rises, and rises when the primary value drops.

Note that you can only specify one port as the synchronous sweep port. This function cannot be used for pulsed sweep or staircase sweep with pulsed bias measurements.

A port address, unit address, or pin number of any measurement pin connected to the synchronous sweep port in one of the Connect_pin functions can be used to specify the synchronous sweep *port*.

The *mode*, *offset*, and *ratio* arguments determine the *start* and *stop* values of the synchronous sweep port as follows:

Mode = POS_SYNC:

$$\text{Start2} = \text{offset} + \text{ratio} \times \text{Start1}$$

$$\text{Stop2} = \text{offset} + \text{ratio} \times \text{Stop1}$$

Mode = NEG_SYNC:

$$\text{Start2} = \text{offset} - \text{ratio} \times \text{Start1}$$

$$\text{Stop2} = \text{offset} - \text{ratio} \times \text{Stop1}$$

Where: Start1 and Stop1 are the *start* and *stop* values of Set_iv (the primary sweep port), and Start2 and Stop2 are the calculated start and stop values of Set_sync (the secondary sweep port).

The *number* of steps in Set_sync is the same as the *number* of steps in Set_iv.

The allowable voltage or current *compliance* you can specify for the synchronous sweep port is determined by the port range setting of synchronous sweep, which in turn is determined by the calculated Start2 and Stop2 values. The following tables list the compliance settings you can specify for each V and I range:

Set_sync (for sweep measurements)

Allowable Compliance for Synchronous Voltage Output Port

Start2/Stop2 Value [V]	V Range	Allowable Compliance		
		HP 41420A	HP 41421B	HP 41424A(VS)
$0 \leq \text{start,stop} \leq 2$	2 V	$\pm(1 \text{ pA to } 1 \text{ A})$	$\pm(1 \text{ pA to } 100 \text{ mA})$	—
$0 \leq \text{start,stop} \leq 14$ $14 < \text{start,stop} \leq 20$	20 V	$\pm(1 \text{ pA to } 1 \text{ A})$ $\pm(1 \text{ pA to } 700 \text{ mA})$	$\pm(1 \text{ pA to } 100 \text{ mA})$ $\pm(1 \text{ pA to } 100 \text{ mA})$	100 mA
$0 \leq \text{start,stop} \leq 40$	40 V	$\pm(1 \text{ pA to } 350 \text{ mA})$	$\pm(1 \text{ pA to } 50 \text{ mA})$	20 mA
$0 \leq \text{start,stop} \leq 100$	100 V	$\pm(1 \text{ pA to } 125 \text{ mA})$	$\pm(1 \text{ pA to } 20 \text{ mA})$	—
$0 \leq \text{start,stop} \leq 200$	200 V	$\pm(1 \text{ pA to } 50 \text{ mA})$	—	—

Allowable Compliance for Synchronous Current Output Port

Start2/Stop2 Value	I Range	Allowable Compliance	
		HP 41420A	HP 41421B
$0 \leq \text{start,stop} \leq 10 \text{ mA}$	1 nA to 10 mA	0 to $\pm 200 \text{ V}$	0 to $\pm 100 \text{ V}$
$0 \leq \text{start,stop} \leq 20 \text{ mA}$ $20 \text{ mA} < \text{start,stop} \leq 50 \text{ mA}$ $50 \text{ mA} < \text{start,stop} \leq 100 \text{ mA}$	100 mA	0 to $\pm 200 \text{ V}$ 0 to $\pm 200 \text{ V}$ 0 to $\pm 200 \text{ V}$	0 to $\pm 100 \text{ V}$ 0 to $\pm 40 \text{ V}$ 0 to $\pm 20 \text{ V}$
$0 \leq \text{start,stop} \leq 125 \text{ mA}$ ¹ $125 \text{ mA} < \text{start,stop} \leq 350 \text{ mA}$ $350 \text{ mA} < \text{start,stop} \leq 700 \text{ mA}$ $700 \text{ mA} < \text{start,stop} \leq 1 \text{ A}$	1 A	0 to $\pm 100 \text{ V}$ 0 to $\pm 40 \text{ V}$ 0 to $\pm 20 \text{ V}$ 0 to $\pm 14 \text{ V}$	—

¹ If start or stop is greater than 100 mA, the 1 A range is selected. If both start and stop are less than or equal to 100 mA, the 100 mA range is selected.

REMAIN specifies that the compliance remain at the present setting unless the unit changes voltage or current source mode by this function. If the specified unit changes voltage or current source status, specifying the REMAIN macro sets the current compliance to 100 μA or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

Example

```
. . .
int err;
. . .
double offset, ratio, comp2;
. . .
err = set_sync(32, POS_SYNC, offset, ratio, comp2, 0.0);
err = sweep_iv(32, I_MEAS, 0.0, meas1, swp1, swp2);
. . .
```

See Also

Set_iv
Set_sync (for sweep measurements)
Sweep_iv
Sweep_miv

Sweep_iv

This function performs staircase sweep, pulsed sweep, staircase sweep with pulsed bias, and synchronous staircase sweep measurements in accordance with parameters established in the Set_iv, Set_piv, Set_pbias and Set_iv, and Set_iv and Set_sync functions, respectively.

Synopsis

```
#include <pcs/tis.h>

int sweep_iv(port, mode, range, measure, source, sync)
int port, mode;
double range, measure[], source[], sync[];
```

For Synchronous Staircase Sweep Measurements:

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	An SWM port address, a pin number, or a unit address that specifies the sweep port. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies whether a voltage measurement or current measurement is performed on the <i>port</i> . You can choose from the following macros: V_MEAS (= 1) I_MEAS (= 2)
<i>range</i> ³	Specifies a measurement range. 0.0 for Auto range mode OR: Numeric expression [A]: -1 to 1 ⁴
<i>measure</i>	A pointer to a double array where the measurement data is stored. Any valid pointer
<i>source</i>	A pointer to a double array where the actual sweep source data is stored. Specify a NULL pointer to discard the sweep source value (<i>sync</i> must also be NULL) or any valid pointer.
<i>sync</i>	A pointer to a double array where the secondary sweep source data is stored. Specify a NULL pointer to discard the results. This argument should be NULL unless the secondary sweep variable is set by the Set_sync function.

- ¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.
- ² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.
- ³ If *mode* is set to V_MEAS, SMUs ignore this argument, and the voltage measurement range is set to the lowest range that includes voltage *compliance* of the measurement port SMU.
- ⁴ Actual range restriction depends on the SMU and SMU port.

For VMs:

Item	Range Restrictions/Description
<i>port</i>	<p>A port address, a pin number, or a unit address that specifies the sweep port.</p> <p><i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹</p> <p><i>unit address</i> 32634, 32636, 32638 . . . 32664</p> <p><i>pin number</i> 1 to 96 ²</p>
<i>mode</i>	<p>Specifies whether a voltage measurement or current measurement is performed on the <i>port</i>. You can choose from the following macros:</p> <p>V_MEAS (= 1) I_MEAS (= 2)</p>
<i>range</i>	<p>Specifies a measurement range. It is not necessary to define the exact range value for each unit.</p> <p>0.0 for Auto range mode OR: Numeric expression [V] Grounded measurements: -40 to 40 Differential measurements: -2 to 2</p>
<i>measure</i>	A pointer to a double array where the measurement data is stored. Any valid pointer
<i>source</i>	<p>A pointer to a double array where the actual sweep source data is stored.</p> <p>Specify a NULL pointer to discard the sweep source value (<i>sync</i> must also be NULL) or any valid pointer.</p>
<i>sync</i>	<p>A pointer to a double array where the secondary sweep source data is stored.</p> <p>Specify a NULL pointer to discard the results.</p> <p>This argument should be NULL unless the secondary sweep variable is set by the Set_sync function.</p>

- ¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.
- ² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VM port.

Sweep_iv

Description

The Sweep_iv function triggers a staircase sweep measurement. The staircase sweep output is determined by the Set_iv function. Measurements are made at the specified measurement port for each sweep step, and the measurement values and sweep source values are returned to the *measure* and *source* arrays, respectively.

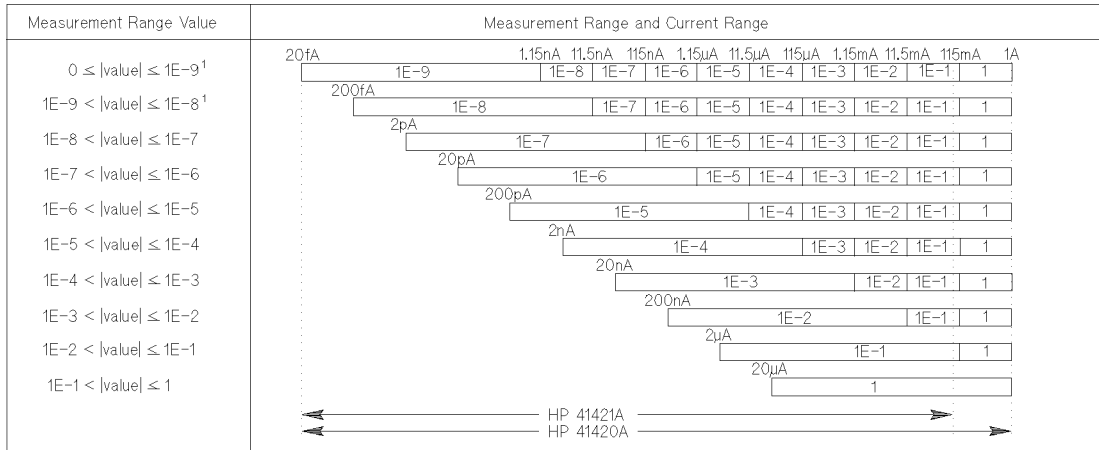
For synchronous staircase sweep measurements, measurements are made at the specified measurement port for each sweep step, and the measurement, the sweep port of Set_iv, and the synchronous sweep port of Set_sync are returned to the *measure*, *source*, and *sync* arrays, respectively.

Note



The *measure*, *source*, and *sync* arrays must have sufficient space to accommodate all return values. If the array size is insufficient, the C Library function may behave unexpectedly without reporting an error since it is not possible to confirm an array size in the C Library. However, this function never affects the extra elements of the value return array, which is larger than the required size. Thus, it is safe and recommended to declare an array size larger than the minimum requirement that the *number* argument for the corresponding Set_iv or Set_piv function determines for these arguments.

The current measurement range of the SMU is determined by the measurement *range* value. The measurement ranges for measurement *mode* = I_MEAS (current measurement by an SMU) are shown in the following figure:



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

For measurement *mode* = I_MEAS, if you specify measurement *range* = 0, Auto range mode is used and the range changes up or down automatically to give the highest resolution. If you specify any other allowed measurement *range*, Limited Auto range mode is used.

For Limited Auto ranging, the range changes between the specified range and all higher ranges automatically. Down ranging to ranges below the initially specified range cannot occur. By specifying a low measurement *range*, a high resolution current measurement can be achieved.

If range changing occurs, the measurement speed decreases because of the SMU wait time and the time required for the range to change.

The voltage measurement range of a VM is determined by the measurement *range* value. The voltage measurement range of an SMU is automatically set to the lowest range that includes voltage *compliance*, set by the Set_iv or Force_i function for the measurement port.

The measurement ranges for measurement *mode* = V_MEAS (voltage measurement by an SMU or VM) are shown in the following figures:

**VM Voltage Ranges for Grounded Measurement Mode
and SMU Voltage Range**

Unit	Voltage Compliance Setting	Measurement Range
HP 41420A	$0 \leq \text{value} \leq 2$	40 μ V 2 V 2
HP 41421B	$2 < \text{value} \leq 20$	400 μ V 20 V 20
	$20 < \text{value} \leq 40$	800 μ V 40 V 40
	$40 < \text{value} \leq 100$	2 mV 100 V 100
HP 41420A	$100 < \text{value} \leq 200$	4 mV 200 V 200
VM (HP 41424A)	Measurement Range Value	40 μ V 2 V 20 V 40 V
	0	2 20 40
	$0 < \text{value} \leq 2$	40 μ V 2 V 2
	$2 < \text{value} \leq 20$	400 μ V 20 V 20
	$20 < \text{value} \leq 40$	800 μ V 40 V 40

VM Voltage Ranges for Differential Measurement Mode

Unit	Measurement Range Value	Measurement Range and Current Range	Maximum Voltage (VM terminal–Circuit Common)
VM (HP 41424A)	0	4 μ V 0.2 V 2 V 4 μ V 0.2 V	Max. 40 V
	$0 < \text{value} \leq 0.2$	4 μ V 0.2 V	
	$0.2 < \text{value} \leq 2$	40 μ V 2 V	

Sweep_iv also gets and returns primary (sweep port) and secondary (synchronous sweep port) values during a synchronous staircase sweep measurement. The DCS returns primary sweep values; secondary sweep values are calculated by Sweep_iv and are stored in the *sync* array. The secondary value calculation method is the same as that performed by the DCS, as follows:

Sweep_iv

V sweep:

sweep mode			range of n	equation
linear sweep		single sweep	1 to N	$V_n = V_{start} + \frac{V_{stop} - V_{start}}{N - 1} (n - 1)$
		double sweep	N + 1 to 2N	$V_n = V_{stop} - \frac{V_{stop} - V_{start}}{N - 1} (n - N - 1)$
logarithmic sweep		single sweep	1 to N	$V_n = V_{start} \times EXP\left(\frac{\log_e\left(\frac{V_{stop}}{V_{start}}\right)}{N - 1} (n - 1)\right)$
		double sweep	N + 1 to 2N	$V_n = V_{stop} \times EXP\left(\frac{\log_e\left(\frac{V_{start}}{V_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

I sweep:

sweep mode			range of n	equation
linear sweep		single sweep	1 to N	$I_n = I_{start} + \frac{I_{stop} - I_{start}}{N - 1} (n - 1)$
		double sweep	N + 1 to 2N	$I_n = I_{stop} - \frac{I_{stop} - I_{start}}{N - 1} (n - N - 1)$
logarithmic sweep		single sweep	1 to N	$I_n = I_{start} \times EXP\left(\frac{\log_e\left(\frac{I_{stop}}{I_{start}}\right)}{N - 1} (n - 1)\right)$
		double sweep	N + 1 to 2N	$I_n = I_{stop} \times EXP\left(\frac{\log_e\left(\frac{I_{start}}{I_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

In Offline Mode:

This function returns simulated measurement values to the *measure* array, to the *statuses* array, and to *source* and *sync*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns source, status, and calculated measurement values. Returned source values are a sequence of linear or logarithmic values in accordance with the *start* and *stop* values specified in the Set_iv or Set_piv function. The status value is always set to 0.

Note that power compliance is not applicable in the offline mode: If *power_compliance* is specified in the Set_iv or Set_piv function, it is ignored.

Calculated measurement values are determined by the following equation:

$$nth \text{ measurement value} = \frac{\text{measurement value}}{\text{number of steps} - 1} \times (n - 1)$$

Where *number* of steps is set by the Set_iv or Set_piv function, and measurement value is determined as follows:

Current Measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	= 0	I compliance	Same as SMU output value
	≠ 0	Range value	
VS	any	20 mA	Same as VS output value

Voltage Measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	≠ 0	Range value	Same as measurement range

File Data Simulation mode:

This function reads <meas>, <status>, and <source> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Sweep_iv
[
<meas> <status> <source>
<meas> <status> <source>
. . . . .
<meas> <status> <source>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Sweep_iv

For Pulsed Sweep Measurements:

Arguments

For SMUs:

Item	Range Restrictions/Description
<i>port</i>	An SWM port address, a pin number, or a unit address that specifies the sweep port. <i>port address</i> 32701 to 32704, 32729, 32732 ¹ <i>unit address</i> 32621 to 32632 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies whether a voltage measurement or current measurement should be performed on the <i>port</i> . You can choose from the following macros: V_MEAS (= 1) I_MEAS (= 2)
<i>range</i> ³	Specifies a measurement range. 0.0 for Auto range mode OR: Numeric expression [A]: -1 to 1 ⁴
<i>measure</i>	A pointer to a double array where the measurement data is stored. Any valid pointer
<i>source</i>	A pointer to a double array where the actual sweep source data is stored. Specify a NULL pointer to discard the sweep source value (<i>sync</i> must also be NULL). Any valid pointer
<i>sync</i>	A pointer to a double array where the secondary sweep source data is stored. Specify a NULL pointer to discard the results. This argument should be NULL unless the secondary sweep variable is set by the Set_sync function.

¹ If 32729 or 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired SMU port.

³ If the measurement *mode* is set to V_MEAS, SMUs ignore this argument, and the voltage measurement range is set to the lowest range that includes voltage *compliance* of the measurement port SMU.

⁴ Actual range restriction depends on the SMU and SMU port.

For VMs:

Item	Range Restrictions/Description
<i>port</i>	A port address, a pin number, or a unit address that specifies the sweep port. <i>port address</i> 32705, 32706, 32708, 32709, 32721 to 32728, 32730, 32731 ¹ <i>unit address</i> 32634, 32636, 32638 . . . 32664 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies whether a voltage measurement or current measurement should be performed on the <i>port</i> . Choose from the following macros: V_MEAS (= 1)
<i>range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: Numeric expression [V]: -40 to 40
<i>measure</i>	A pointer to a double array where the measurement data is stored. Any valid pointer
<i>source</i>	A pointer to a double array where the actual sweep source data is stored. Specify a NULL pointer to discard the sweep source value (<i>sync</i> must also be NULL). Any valid pointer
<i>sync</i>	A pointer to a double array where the secondary sweep source data is stored. Specify a NULL pointer to discard the results. This argument should be NULL unless the secondary sweep variable is set by the Set_sync function.

¹ If 32721 to 32728, 32730, or 32731 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired VM port.

Description

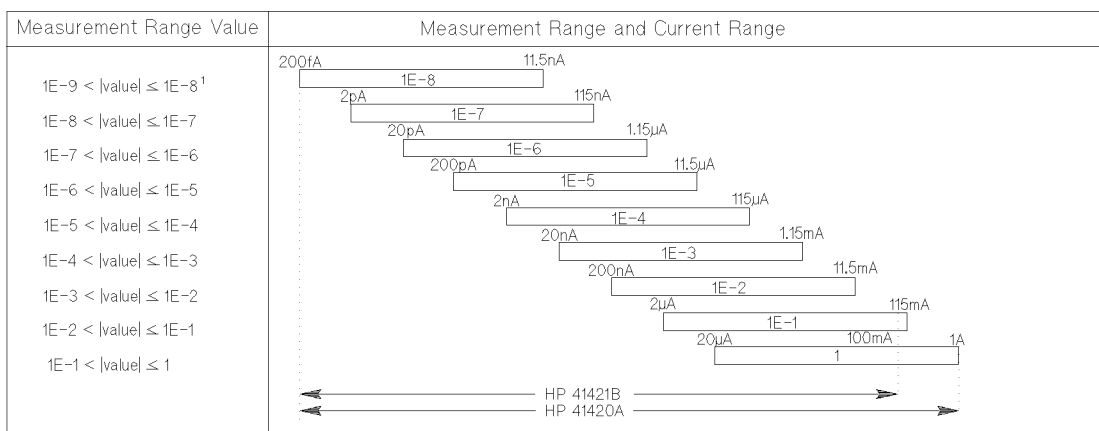
The Sweep_iv function triggers a pulsed sweep measurement or a staircase sweep with pulsed bias measurement with source output determined by the Set_piv program or by the Set_pbias and Set_iv functions. Measurements are made at the specified measurement port for each sweep step, and the measurement values and sweep source values are returned to the *measure* array and the *source* array, respectively.

For staircase sweep with pulsed bias measurements, a staircase sweep is output as determined by the Set_iv function. For each sweep step, a pulse is output as determined by the Set_pbias function, and the measurement is made at the specified measurement port. For pulsed sweep measurements, a pulsed sweep is output as determined by the Set_piv function, and measurements are made at the specified measurement port for each sweep step.

Note

The *measure*, *source*, and *sync* arrays must have sufficient space to accommodate all return values. If the array size is insufficient, the C Library function may behave unexpectedly without reporting an error since it is not possible to confirm an array size in the C Library. However, this function never affects the extra elements of the value return array, which is larger than the required size. Thus, it is safe and recommended to declare an array size larger than the minimum requirement that the *number* argument for the corresponding Set_iv or Set_piv function determines for these arguments.

The current measurement range of an SMU is determined by the measurement *range* value. The measurement ranges for measurement *mode* = I_MEAS (current measurements) are shown in the following figure:



¹ When using an SMU connected via the SMU1 port, you can specify a value less than 1E-8 (10 nA) because measurement value accuracy at ranges below the 1E-8 current range is guaranteed only at the SMU1 port.

The voltage measurement range of a VM is determined by the measurement *range* value. The voltage measurement range of an SMU is automatically set to the lowest range that includes voltage *compliance*, which is set by the Set_iv, Set_piv, Set_pbias, or Force_i functions for the measurement port.

The measurement ranges for measurement *mode* = V_MEAS (voltage measurement—SMU or VM) are shown in the following figure:

Unit	Voltage Compliance Setting	Measurement Range
HP 41420A	$0 \leq value \leq 2$	40 μ V 2 V 2
HP 41421B	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40
	$40 < value \leq 100$	2 mV 100 V 100
HP 41420A	$100 < value \leq 200$	4 mV 200 V 200
VM (HP 41424A)	Measurement Range Value	800 μ V 40 V 40
	0	
	$0 < value \leq 2$	40 μ V 2 V 2
	$2 < value \leq 20$	400 μ V 20 V 20
	$20 < value \leq 40$	800 μ V 40 V 40

In Offline Mode:

This function returns simulated measurement values to the *measure* array, to the *statuses* array, and to *source* and *sync*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns source, status, and calculated measurement values. Returned source values are a sequence of linear or logarithmic values according to the *start* and *stop* values specified in the Set_iv or Set_piv function. Status value is always set to 0.

Note that power compliance is not applicable in the offline mode: If *power_compliance* is specified in the Set_iv or Set_piv function, it is ignored.

Calculated measurement values are determined by the following equation:

$$nth \text{ measurement value} = \frac{\text{measurement value}}{\text{number of steps} - 1} \times (n - 1)$$

Where: *Number* of steps is set by the Set_iv or Set_piv function, and measurement value is determined as follows:

For current measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	= 0	I compliance	Same as SMU output value
	≠ 0	Range value	
VS	any	20 mA	Same as VS output value

Sweep_iv

For voltage measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	≠ 0	Range value	Same as measurement <i>range</i>

File Data Simulation mode:

This function reads <meas>, <status>, and <source> from the data simulation file and returns the values. The data simulation file format is as follows:

```
Sweep_iv
[
<meas> <status> <source>
<meas> <status> <source>
. . .
<meas> <status> <source>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . .
int err;
. . .
double offset, ratio, comp2;
. . .
err = set_sync(32, POS_SYNC, offset, ratio, comp2, 0.0);
err = sweep_iv(32, I_MEAS, 0.0, meas1, swp1, swp2);
. . .
. . .
```

See Also

Rsweep_iv
Rsweep_miv
Set_iv
Set_sync (for sweep measurements)
Sweep_miv

Sweep_miv

This function performs multichannel staircase sweep and multichannel synchronous staircase sweep measurements using up to eight ports at a time. Note that this function cannot be used for pulsed sweep or staircase sweep with pulsed bias measurements.

Synopsis

```
#include <pcs/tis.h>

int sweep_miv(n, ports, ranges, measures, source, sync)
int n, ports[n];
double ranges[n], measures[n] [], source[], sync[];
```

Arguments

For Multichannel Staircase Sweep Measurements:

Item	Range Restrictions/Description
<i>n</i>	Specifies a number of ports where the sweep measurement is performed. This value determines the size of the following arrays: <i>ports</i> , <i>ranges</i> , and the primary index of <i>measures</i> . 1 through 8
<i>ports</i>	Specifies a pointer to an integer array that contains a list of SWM port addresses, pin numbers, or unit addresses. Must have at least <i>n</i> elements.
<i>ranges</i>	Specifies a pointer to a double float array in which the element contains a measurement range on the corresponding port. Must have at least <i>n</i> elements. 0.0 for Auto range mode.
<i>measures</i>	A pointer to a two dimensional array where the measurement values are stored. Primary subscript corresponds to each measurement port (<i>n</i>). Primary subscript $\geq n$. Secondary subscript corresponds to each sweep point (<i>number</i> in Set_iv). Secondary = Set_iv <i>number</i> .
<i>source</i>	A pointer to a double array where the actual sweep source data is stored. Specify NULL to discard the sweep source value (<i>sync</i> must also be NULL). Any valid pointer
<i>sync</i>	A pointer to a double array where the secondary sweep source data is stored. Specify NULL to discard the results. This argument should be NULL unless the secondary sweep variable is set by the Set_sync function.

Sweep_miv

Description

This function performs multichannel staircase sweep measurements according to the measurement parameters set in the Set_iv function or in the Set_iv and Set_sync functions. Measurements are made at the ports specified in the measurement *ports* array of this function. Measurement and sweep source values are returned to the *measures* and *source* arrays, respectively.

This function performs up to 8 simultaneous sweep measurements at once. The first argument *n* specifies the number of ports to measure. The valid range of *n* is 1 through 8; if only one port is required, the Sweep_iv function should be used.

Note that this procedure assumes the measurement value return array size to be defined exactly $n \times \text{points}$, where *n* is the number of simultaneous measurement ports and *points* is the number of measurement points specified by the Set_iv function. If the actual size (especially of the secondary index) does not match the size of *points*, you may get unexpected results. In the worst case, a segmentation violation, a bus error, or a violation on content of an irrelevant variable may occur.

The pulsed sweep measurement of pulse biased sweep is not supported by this procedure.

The measurement *ports* array contains the port address, unit address, or pin number lists of the specified measurement ports. If you assign port addresses, unit addresses, or pin numbers to the measurement *ports* array, the ports indicated by the port addresses and the measurement pins indicated by the pin numbers must be connected to the measurement units before this function executes.

You can specify the same measurement port in the measurement *ports* array. Note that the actual number of measurement ports is less than the number of specified measurement ports.

You must specify the measurement ports in ascending order in the *ports* array. The measurement mode used is the same as the previous measurement mode set for the measurement port.

Sweep_miv also obtains and returns primary and secondary values during a multichannel synchronous staircase sweep measurement. The DCS returns primary sweep values; secondary sweep values are calculated by Sweep_miv and are stored in the *sync* array. Secondary value calculation method is the same as that performed by the DCS, as follows:

V sweep:

sweep mode		range of n	equation
linear sweep	single sweep	1 to N	$V_n = V_{start} + \frac{V_{stop} - V_{start}}{N - 1} (n - 1)$
	double sweep	N + 1 to 2N	$V_n = V_{stop} - \frac{V_{stop} - V_{start}}{N - 1} (n - N - 1)$
logarithmic sweep	single sweep	1 to N	$V_n = V_{start} \times EXP\left(\frac{\log_e\left(\frac{V_{stop}}{V_{start}}\right)}{N - 1} (n - 1)\right)$
	double sweep	N + 1 to 2N	$V_n = V_{stop} \times EXP\left(\frac{\log_e\left(\frac{V_{start}}{V_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

I sweep:

sweep mode		range of n	equation
linear sweep	single sweep	1 to N	$I_n = I_{start} + \frac{I_{stop} - I_{start}}{N - 1} (n - 1)$
	double sweep	N + 1 to 2N	$I_n = I_{stop} - \frac{I_{stop} - I_{start}}{N - 1} (n - N - 1)$
logarithmic sweep	single sweep	1 to N	$I_n = I_{start} \times EXP\left(\frac{\log_e\left(\frac{I_{stop}}{I_{start}}\right)}{N - 1} (n - 1)\right)$
	double sweep	N + 1 to 2N	$I_n = I_{stop} \times EXP\left(\frac{\log_e\left(\frac{I_{start}}{I_{stop}}\right)}{N - 1} (n - N - 1)\right)$

n: the n-th measurement point

N: number of steps determined in Set_iv subprogram

For more information on these values, refer to the description of the Sweep_iv function for synchronous staircase sweep measurements.

In Offline Mode:

This function returns simulated measurement values to the *measures* array, to the *statuses* array, and to *source* and *sync*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns source, status, and calculated measurement values. Returned source values are a sequence of linear or logarithmic values in accordance with the *start* and *stop* values specified in the Set_iv or Set_piv function. Status value is always set to 0.

Note that power compliance is not applicable in the offline mode: If *power_compliance* is specified in the Set_iv or Set_piv function, it is ignored.

Calculated measurement values are determined by the following equation:

$$nth \text{ measurement value} = \frac{\text{measurement value}}{\text{number of steps} - 1} \times (n - 1)$$

Where *number of steps* is set by the Set_iv or Set_piv function, and measurement value is determined as follows:

Sweep_miv**For current measurements**

Unit	Meas. Range	Measurement Value	Polarity
SMU	= 0	I compliance	Same as SMU/VS output value
	≠ 0	Range value	
VS	any	20 mA	

For voltage measurements

Unit	Meas. Range	Measurement Value	Polarity
SMU	—	V compliance	Same as SMU output value
VM	= 0	2.0 V	Always positive
	≠ 0	Range value	Same as measurement <i>range</i>

File Data Simulation mode:

This function reads <meas>, <status>, <source1>, and <source2> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Sweep_miv
[
<meas1> <meas2> . . . <st1> <st2> . . . <source1> <source2>
<meas1> <meas2> . . . <st1> <st2> . . . <source1> <source2>
. . . . .
<meas1> <meas2> . . . <st1> <st2> . . . <source1> <source2>
]
```

If you do not specify these values, the constant data simulation mode values are used. You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . . .
int err;
. . . .
int n=3, ports;
double ranges[3], measures[3][101], sweep[3];
. . . .
ports[0] = 8; ports[1] = 12; ports[2] = 16;
ranges[0] = 0.0; ranges[1] = 0.0; ranges[2] = 0.0;
. . . .
err = sweep_miv(n, ports, ranges, measures, sweep, NULL);
. . . .
```

See Also

R sweep_iv
 R sweep_miv
 Set_iv
 Set_sync (for sweep measurements)
 Sweep_iv

Search Measurements

Introduction

The C Library provides the following functions for search measurements:

- Search_iv
- Set_asearch
- Set_bsearch
- Set_lsearch
- Set_sync

These functions set the parameters for analog, binary, or linear searches, and synchronize an SMU with a search port. The measurements are performed for feedback information.

The functions are arranged alphabetically in this chapter.

Search_iv

This function performs search measurements according to the conditions established by the Set_asearch, Set_bsearch, or Set_lsearch functions, and returns the resultant search value.

If you include a Set_sync function between Set_bsearch or Set_lsearch and the Search_iv function, Search_iv performs a search measurement using the synchronous search unit specified in Set_sync.

Synopsis

```
#include <pcs/tis.h>

int search_iv(search, status, sense)
int *status;
double *search, *sense;
```

Arguments

Item	Range Restrictions/Description
<i>search</i>	A pointer to a double where the searched value is stored. Any valid pointer
<i>status</i>	A pointer to an integer where the search measurement status is stored. One of the following statuses is returned for this argument: SEARCH_DOMAIN (= 1) SEARCH_ABNORMAL (= 2) SEARCH_FAILED (= 4)
<i>sense</i>	A pointer to a double where the sense port value at the convergence is returned. Any valid pointer

Description

This function performs a search measurement and returns the search value to a user-specified variable. If a measurement does not execute normally or if the search value is not found, the value -9999999.99999 is returned.

If the value -7777777.77777 is returned, change the minimum search value so the difference between the *min* and *max* search values is larger. If the value -8888888.88888 is returned, change the maximum search value so the difference between the *min* and *max* search values is larger.

The Search_iv function returns a macro in which bit masks are used to test each bit of this parameter, by the bit AND (&) operator.

The following lists the macros returned for each search function and their meanings.

Set_asearch:

SEARCH_DOMAIN : Search value not found (between the *start* and *stop* values).

SEARCH_ABNORMAL : Abnormal data (for example, oscillating).

SEARCH_FAILED : Parameter(s) incorrectly set for this search (search value not found).

Set_bsearch:

SEARCH_DOMAIN : Search value not found (between the *min* and *max* search values).¹

SEARCH_ABNORMAL : Abnormal data.

SEARCH_FAILED : Sense value is not within the defined error of target value.

Set_lsearch:

SEARCH_DOMAIN : Search value not found (between the *start* and *stop* values).

SEARCH_ABNORMAL : Abnormal data

SEARCH_FAILED : Abnormal compliance loop

A Force_i or Force_v statement cannot be used between the Search_iv and Set_asearch, Set_bsearch, or Set_lsearch statements.

In Offline Mode:

This function returns simulated measurement values to the *search*, *status*, and *sense* arguments. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

For Analog Feedback Measurements:

Search value is set to the average value of the *start* and *stop* values in the Set_asearch function. *Status* is always set to 0, which in the offline mode means that the search value is found.

Search_iv

For Binary Search Measurements:

Search value is set to the average value as *min* and *max* in the Set_bsearch function. *Status* is always set to 0, which in the offline mode means that the search value was found.

For Linear Search Measurements:

Search value is set to the average value of the *start* and *stop* values in the Set_lsearch function. *Status* is always set to 0, which in the offline mode means that the search value is found.

File Data Simulation mode:

This function reads <search>, <status>, and <sense> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Search  <search>  <status>  <sense>
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . .
int drain, gate, source, stat;
double search;
. . .
. . .
if (set_asearch(gate, drain, NEG_FEEDBACK, VS_IM, 0.0,
               5.0, -10.0, 1e-5, 500.0,
               1e-6, -0.05, 0.1, 0.1, 0.1);
    == -1) error_rep();
if (search_iv(&search, &stat, NULL) == -1) error_rep();
. . .
. . .
```

See Also

Set_asearch
Set_bsearch
Set_lsearch
Set_sync (for search measurements)

Set_asearch

This function sets the parameters for analog search measurements.

Synopsis

```
#include <pcs/tis.h>

int set_asearch(search_port, sense_port,
                search_mode, meas_mode,
                start, stop, sense_bias,
                target, ramp_rate,
                search_comp, sense_comp,
                hold, delay, integ_time)
int search_port, sense_port, search_mode, meas_mode;
double start, stop, sense_bias;
double target, ramp_rate, search_comp, sense_comp;
double hold, delay, integ_time;
```

Arguments

Item	Range Restrictions/Description
<i>search_port</i>	Specifies an SWM port address, a pin number currently connected to a port, or an SMU unit address.
	<i>port address</i> 32701 to 32704, 32729 ⁵ , 32732 ⁵
	<i>SMU address</i> 32621 to 32632
	<i>pin number</i> 1 to 96 ¹
<i>sense port</i>	Specifies an SWM port address, a pin number currently connected to a port, or a DCS unit address.
	<i>port address</i> 32701 to 32704, 32729 ⁵ , 32732 ⁵
	<i>SMU address</i> 32621 to 32632
	<i>pin number</i> 1 to 96 ¹
<i>search_mode</i>	Specifies the type of analog search. Choose from one of the following macros: NEG_FEEDBACK (= 1) POS_FEEDBACK (= 2) POS_RAMP (= 3) NEG_RAMP (= 4)

Set_asearch

Item	Range Restrictions/Description
<i>meas_mode</i>	<p>Specifies whether voltage or current should be measured on the <i>search_port</i> or <i>sense_port</i>. This argument also determines if the sensed value at the converged point is returned by the Search_iv function. You can choose from the following macros (see the Description section for the definitions):</p> <p>IS_VM (= -2) IS_VM_MEAS (= -4) IS_IM (= 2) IS_IM_MEAS (= 4) VS_VM (= -1) VS_VM_MEAS (= -3) VS_IM (= 1) VS_IM_MEAS (= 3)</p>
<i>start, stop</i>	<p>Specifies the start and stop voltage on the <i>search_port</i>.</p> <p>Numeric expression [V]: -200 to 200 ^{2, 3}</p>
<i>sense_bias</i>	<p>Specifies the voltage or current force value on the <i>sense_port</i> during the search measurement.</p> <p>Sense SMU in VS mode [V]: -200 to 200 ³ Sense SMU in IS mode [A]: -1 to 1 ³</p>
<i>target</i>	<p>Specifies the target value at the <i>sense_port</i>.</p> <p>Sense SMU in VS mode [A]: -1 to 1 ³ Sense SMU in IS mode [V]: -200 to 200 ³</p>
<i>ramp_rate</i>	<p>Specifies the ramp rate of the search measurement.</p> <p>Specify 0.0 for 500 V/s OR: 0.5 to 50000 V/s ²</p>
<i>search_comp, sense_comp</i>	<p>Specifies the voltage or current compliance on either <i>search_port</i> or <i>sense_port</i>.</p> <p>REMAIN macro OR: Sense SMU in VS mode: -1 to 1 ³ Sense SMU in IS mode: -200 to 200 ³</p>
<i>hold</i>	<p>Specifies the hold time between search trigger and actual search start.</p> <p>Numeric expression [s]: 0 to 65.535 Resolution: 0.001</p>
<i>delay</i>	<p>Specifies the measurement delay time between the actual measurement and the search convergence.</p> <p>Numeric expression [s]: 0 to 65.535 Resolution: 0.001</p>
<i>integ_time</i>	<p>Specifies the analog feedback loop integration time.</p> <p>This value is not relevant to the dc source measurement integration time specified by the Set_smu function.</p> <p>Specify 0.0 for 5.0E-3 second default OR: Numeric expression: [s] 5.0E-7 to 0.45 ⁴</p>

7-6 Search Measurements

- ¹ 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a connected pin to the desired port.
- ² This range is supported for firmware revision 3.00 or later in the DCS. Otherwise, -20 to 20 is available.
- ³ Actual range of these parameters depends on the SMU and SMU port used. Refer to description for details.
- ⁴ This range is supported for firmware revision 3.00 or later in the DCS. Otherwise, $5.0\text{E}-6$ to 0.45 is available.
- ⁵ These port addresses are used for the AUX ports of the port expander.

Description

Set_asearch sets the parameters for analog search measurements. For more on analog search measurements, see “Analog Feedback Unit” in chapter 2 of the *System Information Manual*.

A port address, SMU unit address, or pin number can be used to specify the SMU *port*.

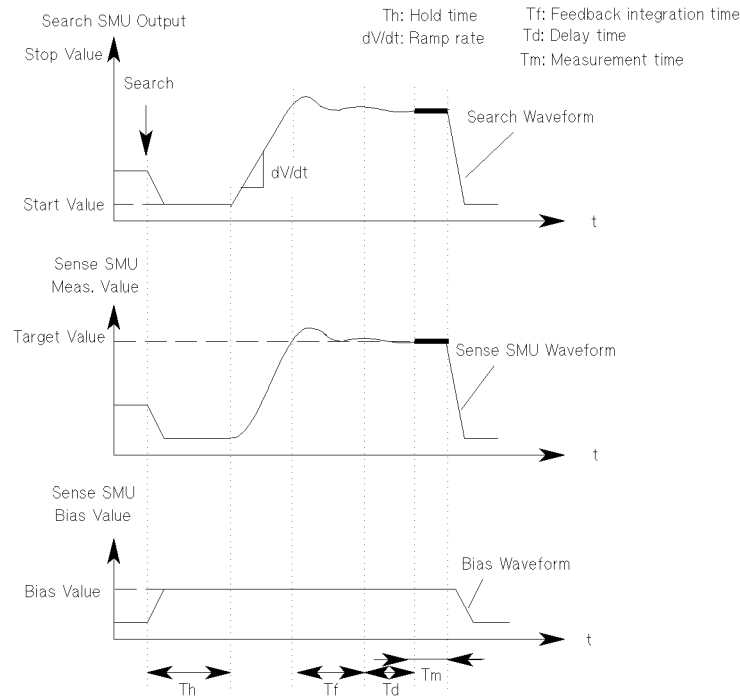
Note that the *meas_mode* does not affect the VS mode of the search port. The search is always in the voltage mode, regardless of the specified search port measurement mode. Thus, *start* and *stop* should always be specified as voltage.

Search_modes are explained briefly below:

<i>search mode</i>	Description
NEG_FEEDBACK	Performs a negative feedback analog search. Set this mode if an increase in search value causes an increase in sense value or if a decrease in search value causes a decrease in sense value.
POS_FEEDBACK	Performs a positive feedback analog search. Set this mode if an increase in search value causes a decrease in sense value or if a decrease in search value causes an increase in sense value.
POS_RAMP	Performs a ramp analog search until the sense value is greater than the <i>target</i> . Set this mode if an increase in search value causes an increase in sense value.
NEG_RAMP	Performs a ramp analog search until the sense value is less than the <i>target</i> . Set this mode if an increase in search value causes a decrease in sense value.

The following figure shows an example waveform for *search_mode* = NEG_FEEDBACK:

Set_asearch



The *meas_mode* argument determines the following conditions. If you want to measure the sense SMU value by the *Search_iv* function specified by the *search*, ± 3 or ± 4 must be specified as the *meas_mode*.

<i>meas_mode</i>	Description
IS_VM: -2	I measurement at search port; V measurement at sense port.
IS_VM_MEAS: -4	I measurement at search port; V measurement at sense port; sense value is returned.
IS_IM: 2	I measurement at search port; I measurement at sense port.
IS_IM_MEAS: 4	I measurement at search port; I measurement at sense port; sense value is returned.
VS_VM: -1	V measurement at search port; V measurement at sense port.
VS_VM_MEAS: -3	V measurement at search port; V measurement at sense port; sense value is returned.
VS_IM: 1	V measurement at search port; I measurement at sense port.
VS_IM_MEAS: 3	V measurement at search port; I measurement at sense port; sense value is returned.

The *start* value can be less or greater than the *stop* value, but they cannot be equal. The output voltage range in the search SMU is automatically set to the range that can output the greater absolute *start* or *stop* value with the highest resolution, but these two arguments must satisfy the following conditions:

7-8 Search Measurements

Output Voltage Range in Search SMU	<i>start</i> and <i>stop</i> value conditions
2 V	$0 < \text{MAX}(\text{start} , \text{stop}) \leq 2 \text{ V}$, $0.1 \text{ V} \leq \text{start} - \text{stop} \leq 2 \text{ V}$
20 V	$2 \text{ V} \leq \text{MAX}(\text{start} , \text{stop}) \leq 20 \text{ V}$, $1 \text{ V} \leq \text{start} - \text{stop} \leq 20 \text{ V}$
40 V ¹	$20 \text{ V} \leq \text{MAX}(\text{start} , \text{stop}) \leq 40 \text{ V}$, $2 \text{ V} \leq \text{start} - \text{stop} \leq 40 \text{ V}$
100 V ¹	$40 \text{ V} \leq \text{MAX}(\text{start} , \text{stop}) \leq 100 \text{ V}$, $5 \text{ V} \leq \text{start} - \text{stop} \leq 100 \text{ V}$
200 V ¹	$100 \text{ V} \leq \text{MAX}(\text{start} , \text{stop}) \leq 200 \text{ V}$, $10 \text{ V} \leq \text{start} - \text{stop} \leq 200 \text{ V}$

¹ These ranges are supported for firmware revision 3.00 or later in the DCS (HP 4142B).

Sense_bias specifies the voltage or current force value on the *sense_port*. The forcing is operated only during the search measurement. In other words, the force value on the *sense_port* is set to this value when the search is triggered, and reset to the previous value after the search is done.

The range allowed for the *sense_bias*, *target*, *search_comp*, and *sense_comp* arguments depend on each SMU range and the allowable voltage or current of each SMU port. Refer to Specifications for details. Set the *target* to less than 90% of the *sense_comp* value.

Ramp_rate and *integ_time* depend on the output voltage range in the search SMU as shown in the following table:

Output Voltage Range in Search SMU	Available Ramp Rate (R) [V/s]	Available Feedback Integration Time (t)
2 V	$0.5 \leq R \leq 5000$	$50 \mu\text{s} \leq t \leq 450 \text{ ms}$
20 V	$5.5 \leq R \leq 50000$	$5 \mu\text{s} \leq t \leq 45 \text{ ms}$
40 V ¹	$10 \leq R \leq 100000$	$2.5 \mu\text{s} \leq t \leq 25 \text{ ms}$
100 V ¹	$25 \leq R \leq 100000$	$1 \mu\text{s} \leq t \leq 10 \text{ ms}$
200 V ¹	$55 \leq R \leq 100000$	$0.5 \mu\text{s} \leq t \leq 5 \text{ ms}$

¹ These ranges are supported for firmware revision 3.00 or later of the DCS (HP 4142B).

Feedback *integ_time* resolutions are shown in the following table:

Set_asearch

Feedback Integration Time (t)	Setting Resolution
$0.5 \mu\text{s} \leq t \leq 4.5 \mu\text{s}$	$0.5 \mu\text{s}$
$5 \mu\text{s} \leq t \leq 45 \mu\text{s}$	$5 \mu\text{s}$
$50 \mu\text{s} \leq t \leq 450 \mu\text{s}$	$50 \mu\text{s}$
$500 \mu\text{s} \leq t \leq 4.5 \text{ ms}$	$500 \mu\text{s}$
$5 \text{ ms} \leq t \leq 45 \text{ ms}$	5 ms
$50 \text{ ms} \leq t \leq 450 \text{ ms}$	50 ms

Ramp_rate resolutions (unit: V/s) are shown in the following table:

Ramp Rate (R)	Voltage Range in Search SMU				
	2 V range	20 V range	40 V range	100 V range	200 V range
$0.5 \text{ V/s} \leq R \leq 5 \text{ V/s}$	0.05	—	—	—	—
$5.5 \text{ V/s} \leq R \leq 10 \text{ V/s}$	0.5	0.5	—	—	—
$10 \text{ V/s} \leq R \leq 25 \text{ V/s}$	0.5	0.5	1	—	—
$25 \text{ V/s} \leq R \leq 50 \text{ V/s}$	0.5	0.5	1	2.5	—
$55 \text{ V/s} \leq R \leq 100 \text{ V/s}$	5	5	5	5	5
$100 \text{ V/s} \leq R \leq 250 \text{ V/s}$	5	5	10	5	5
$250 \text{ V/s} \leq R \leq 500 \text{ V/s}$	5	5	10	25	5
$550 \text{ V/s} \leq R \leq 1000 \text{ V/s}$	50	50	50	50	50
$1000 \text{ V/s} \leq R \leq 2500 \text{ V/s}$	50	50	100	50	50
$2500 \text{ V/s} \leq R \leq 5000 \text{ V/s}$	50	50	100	250	50
$5500 \text{ V/s} \leq R \leq 10000 \text{ V/s}$	—	500	500	500	500
$10000 \text{ V/s} \leq R \leq 25000 \text{ V/s}$	—	500	1000	500	500
$25000 \text{ V/s} \leq R \leq 50000 \text{ V/s}$	—	500	1000	2500	500
$55000 \text{ V/s} \leq R \leq 100000 \text{ V/s}$	—	—	5000	5000	5000

Hold specifies the hold time between search trigger and actual search start. After the measurement is triggered by the Search_iv function, the forcing voltage on the *search_port* starts rising (or dropping) after the number of seconds you specify for this argument elapse.

Example

```

. . .
int drain, gate, source, stat;
double search;
. . .
if (set_asearch(gate, drain, NEG_FEEDBACK, VS_IM, 0.0, 5.0,
               -10.0, 1e-5, 500, 1e-6,
               2e-5, 0.1, 0.1, 0.1)
    == -1) error_rep();
if (search_iv(&search, &stat, NULL) == -1) error_rep();
. . .

```

See Also

[Search_iv](#)
[Set_bsearch](#)
[Set_lsearch](#)
[Set_sync](#) (for search measurements)

Set_bsearch

This function sets the parameters for binary search measurements.

Synopsis

```
#include <pcs/tis.h>

int set_bsearch(search_port, sense_port, mode,
                min, max, target, range,
                search_comp, hold, conv_condition)
int search_port, sense_port, mode;
double min, max, target, range;
double search_comp, hold, conv_condition;
```

Arguments

Item	Range Restrictions/Description
<i>search_port</i>	Specifies an SWM port address, a pin number currently connected to a port, or a DCS unit address. <i>port address</i> 32701 to 32706, 32708, 32709, 32721 to 32732 ¹ <i>unit address</i> 32621 to 32633, 32635, 32637, 32639, . . . 32663 <i>pin number</i> 1 to 96 ²
<i>sense port</i>	Specifies an SWM port address, a pin number currently connected to a port, or a DCS unit address. <i>port address</i> 32701 to 32706, 32708, 32709, 32721 to 32732 ¹ <i>unit address</i> 32621 to 32632, 32634, 32636, 32638, . . . 32664 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies the VS/IS mode on the <i>search_port</i> , and the NORMAL/RESET mode and ACCURACY/TIMES mode of the binary search. Choose from the following macros for this argument (see the Description section for the definitions): VS_NORM_ACC (= 0) IS_NORM_ACC (= 1) VS_RESET_ACC (= 2) IS_RESET_ACC (= 3) VS_NORM_TIMES (= 4) IS_NORM_TIMES (= 5) VS_RESET_TIMES (= 6) IS_RESET_TIMES (= 7)

Item	Range Restrictions/Description
<i>min, max</i>	Specifies the initial lower and upper boundaries of the binary search. Search unit is SMU in VS mode [V]: -200 to 200 ³ Search unit is VS [V]: -40 to 40 Search unit is SMU in IS mode [A]: -1 to 1 ⁴
<i>target</i>	Specifies the target value at <i>sense_port</i> . Sense unit is SMU in VS mode [A]: -1 to 1 ⁴ Sense unit is VM [V]: -40 to 40 Sense unit is SMU in IS mode [V]: -200 to 200 ³
<i>range</i>	Specifies a measurement range. 0.0 for Auto range mode OR: Sense unit is SMU in VS mode [A]: -1 to 1 Sense unit is VM [V]: -40 to 40 Sense unit is SMU in IS mode [V]: any value ⁵
<i>search_comp</i>	Specifies the voltage or current compliance on <i>search_port</i> . REMAIN macro OR: Search unit is SMU in VS mode [A]: -1 to 1 ³ Search unit is VS: any value ⁵ Search unit is SMU in IS mode [V]: -200 to 200 ⁴
<i>hold</i>	Specifies a settling wait time between forcing the <i>search_port</i> output and the measurement. In RESET mode, the same amount of wait time is taken each time the <i>search_port</i> is set to the <i>min</i> value. Numeric expression [s]: 0 to 6.5535
<i>conv_condition</i>	If the ACCURACY mode is selected in <i>mode</i> , this argument specifies the error range of the convergence. If the TIMES mode is selected by <i>mode</i> , this argument specifies the number of iterations. Pass a double float to this argument, even if this argument is specifying the number of iterations (an integer). Accuracy mode [%]: 0 to 100 ⁷ Resolution: 0.01 Number of times mode: 1 to 16

¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ Maximum output voltage and actual range restrictions for current compliance depend on the specified voltage range.

⁴ Actual range restrictions for voltage compliance depend on the specified test current.

⁵ These are dummy parameters.

⁶ If the absolute *min* or *max* search value is larger than 100 mA, default is set to 2 V.

⁷ If the specified convergence accuracy is 0%, actual accuracy is set to 0.01%.

Set_bsearch

Description

Set_bsearch divides the set of items to be searched into two equal parts, then determines which part contains the *target*. This process repeats until the specified *conv_condition* is satisfied.

A pin number or port address can be used to specify a unit. The *mode* argument determines the following conditions:

Mode	Search Port VS/IS Mode	Normal/Reset Mode	Convergence Mode
VS_NORM_ACC	VS	Normal	Accuracy
IS_NORM_ACC	IS	Normal	Accuracy
VS_RESET_ACC	VS	Reset	Accuracy
IS_RESET_ACC	IS	Reset	Accuracy
VS_NORM_TIMES	VS	Normal	Times
IS_NORM_TIMES	IS	Normal	Times
VS_RESET_TIMES	VS	Reset	Times
IS_RESET_TIMES	IS	Reset	Times

Normal and Reset Modes:

In the Normal mode, the output of the search unit increments or decrements directly to the next value. The Reset mode resets the output of the search unit to the minimum value, waits a delay time equal to the hold time, and increments or decrements to the next value.

Convergence mode:

If you select Accuracy mode, convergence is reached if the sense value is within the user defined error range. If the sense value does not converge, even if the output of the search unit is within its resolution, the binary search stops.

If you select Times mode, you must define how many iterations the binary search is to execute. The maximum number of iterations is 16.

The output range of the search port is set to either *max* or *min*, whichever is greater. Refer to the Force_v or Force_i function for output value, range, and compliance relationships.

The *range* should be greater than the *target*. If the *range* is omitted or is less than the *target*, the *range* is set to the *target*.

If you wish to set a compliance for the sense unit, execute a Force_v or Force_i function to set the compliance of the sense unit before you execute a Set_bsearch.

Hold is the time between the output of the search unit and the measurement of the sense unit. If you select a reset *mode*, the hold time is added to the delay time between iterations, and the output returns to the initial value between iterations.

Example

```

. . .
int drain, gate, substrate;
double search;
. . .
. . .
if (force_v(drain, -2.0, -5, 3e-5)
    == -1) error_rep();
if (set_bsearch(gate, drain, VS_NORM_ACC, 0.0, 5.0,
    -1e-5, -2e-5, 0, 0.1)
    == -1) error_rep();
if set_sync(substrate, POS_SYNC, 0.0, NULL, NULL, NULL)
    == -1) error_rep();
if (search_iv(&search, &stat, NULL) == -1) error_rep();
if (disable_port3(gate, drain, substrate)
    == -1) error_rep();
. . .
. . .

```

See Also

Search_iv
 Set_asearch
 Set_lsearch
 Set_sync (for search measurements)

Set_lsearch

This function sets the parameters for a linear search measurement.

Synopsis

```
#include <pcs/tis.h>

int set_lsearch(search_port, sense_port, mode,
                start, stop, step,
                target, range, search_comp, hold)
int search_port, sense_port, mode;
double start, stop, step;
double target, range, search_comp, hold;
```

Arguments

Item	Range Restrictions/Description						
<i>search_port</i>	<p>Specifies an SWM port address, a pin number currently connected to a port, or a DCS unit address.</p> <table><tr><td><i>port address</i></td><td>32701 to 32706, 32708, 32709, 32721 to 32732 ¹</td></tr><tr><td><i>unit address</i></td><td>32621 to 32633, 32635, 32637, 32639, . . . 32663</td></tr><tr><td><i>pin number</i></td><td>1 to 96 ²</td></tr></table>	<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹	<i>unit address</i>	32621 to 32633, 32635, 32637, 32639, . . . 32663	<i>pin number</i>	1 to 96 ²
<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹						
<i>unit address</i>	32621 to 32633, 32635, 32637, 32639, . . . 32663						
<i>pin number</i>	1 to 96 ²						
<i>sense_port</i>	<p>Specifies an SWM port address, a pin number currently connected to a port, or a DCS unit address.</p> <table><tr><td><i>port address</i></td><td>32701 to 32706, 32708, 32709, 32721 to 32732 ¹</td></tr><tr><td><i>unit address</i></td><td>32621 to 32632, 32634, 32636, 32638, . . . 32664</td></tr><tr><td><i>pin number</i></td><td>1 to 96 ²</td></tr></table>	<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹	<i>unit address</i>	32621 to 32632, 32634, 32636, 32638, . . . 32664	<i>pin number</i>	1 to 96 ²
<i>port address</i>	32701 to 32706, 32708, 32709, 32721 to 32732 ¹						
<i>unit address</i>	32621 to 32632, 32634, 32636, 32638, . . . 32664						
<i>pin number</i>	1 to 96 ²						
<i>mode</i>	<p>In the NORMAL mode, the output of the search unit changes directly to the next value.</p> <p>In the RESET mode, the output of the search unit resets to the <i>start</i> value upon each iteration, waits <i>hold</i> seconds, then changes the output to the next value.</p> <p>Choose from one of the following macros for this argument:</p> <p>VS_NORM_ABOVE (= 0) IS_NORM_ABOVE (= 1) VS_RESET_ABOVE (= 2) IS_RESET_ABOVE (= 3) VS_NORM_BELOW (= 4) IS_NORM_BELOW (= 5) VS_RESET_BELOW (= 6) IS_RESET_BELOW (= 7) VS_NORM_STAT (= 8) IS_NORM_STAT (= 9) VS_RESET_STAT (= 10) IS_RESET_STAT (= 11)</p>						

Item	Range Restrictions/Description
<i>start, stop</i>	Specifies the search start and stop values. Search unit is SMU in VS mode [V]: -200 to 200 ³ Search unit is VS [V]: -40 to 40 Search unit is SMU in IS mode [A]: -1 to 1 ³
<i>step</i>	Specifies the increase between each step. Search unit is SMU in VS mode [V]: 0 to 200 ³ Search unit is VS [V]: 0 to 40 Search unit is SMU in IS mode [A]: 0 to 1 ³
<i>target</i>	Specifies the target value at the <i>sense_port</i> . Sense unit is SMU in VS mode [A]: -1 to 1 ³ Sense unit is VM [V]: -40 to 40 Sense unit is SMU in IS mode [V]: -200 to 200 ³
<i>range</i>	Specifies a measurement range at the sense port. 0.0 for Auto range mode OR: Sense unit is SMU in VS mode [A]: -1 to 1 ³ Sense unit is VM [V]: -40 to 40 Sense unit is SMU in IS mode [V]:— ⁴
<i>search_comp</i>	Specifies the voltage or current compliance on <i>search_port</i> . REMAIN macro OR: Search unit is SMU in VS mode [A]: -1 to 1 ³ Search unit is VS:— ⁴ Search unit is SMU in IS mode [V]: -200 to 200 ¹
<i>hold</i>	Specifies a settling wait time between forcing the <i>search_port</i> output and the measurement. In RESET mode, the same amount of wait time is taken each time the <i>search_port</i> is set to the <i>start</i> value. Numeric expression [s]: 0 to 6.5535

¹ If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ Actual *start*, *stop*, *step*, *target*, *range*, and *search_comp* range restrictions depend on the SMU and the SMU port used.

⁴ These are dummy parameters.

Set_Isearch

Description

The *mode* argument determines the following conditions:

Mode	Search Unit VS/IS Mode	Normal/Reset Mode	Stop Mode	Measurement/ Loop Status Mode
VS_NORM_ABOVE	VS	Normal	> target	Measurement
IS_NORM_ABOVE	IS	Normal	> target	Measurement
VS_RESET_ABOVE	VS	Reset	> target	Measurement
IS_RESET_ABOVE	IS	Reset	> target	Measurement
VS_NORM_BELOW	VS	Normal	< target	Measurement
IS_NORM_BELOW	IS	Normal	< target	Measurement
VS_RESET_BELOW	VS	Reset	< target	Measurement
IS_RESET_BELOW	IS	Reset	< target	Measurement
VS_NORM_STAT	VS	Normal	> target	Loop status
IS_NORM_STAT	IS	Normal	> target	Loop status
VS_RESET_STAT	VS	Reset	> target	Loop status
IS_RESET_STAT	IS	Reset	> target	Loop status

Note



Set the sense unit to VS mode if you set the *mode* argument to one of the four loop status modes.

Normal and Reset Modes:

In the Normal mode, the output of the search unit increments or decrements directly to the next value.

The Reset mode resets the output of the search unit to the start value, waits a delay time equal to the hold time, then increments or decrements to the next value.

Stop Mode:

Terminates a search when the sense value is greater than or less than the *target*, as determined by the *mode* setting.

Measurement/Loop Status Mode:

The Measurement or Loop Status is determined by whether or not the sense value has reached the *target*. The measurement status mode compares the measurement data with the target value and stops the measurement when the target value is greater than or equal to the measurement value. The loop status mode compares the target value with compliance and when the two values equal, the measurement stops.

The *start* value can be less than or greater than the *stop* value, but they cannot be equal. The *step* value must be positive. If the *start* value is less than the *stop* value, the *step* value is

added to the *start* value at each iteration. If the *start* value is greater than the *stop* value, the *step* value is subtracted at each iteration.

The output range of the search unit is set to the greater absolute *start* or *stop* value. Refer to the Force_v or Force_i function for output value, range, and compliance relationships.

The measurement *range* should be greater than the *target*. If the measurement *range* is less than the *target*, the measurement *range* is set to the *target*.

If you wish to set a compliance for the sense unit, execute a Force_v or Force_i function to set the compliance of the sense unit before you execute a Set_lsearch.

The output of the sense unit is set to the *start* value after a feedback measurement executes.

Example

```
. . .
int err;
. . .
err = set_lsearch(base, collector, IS_RESET_ABOVE, start,
                  stop, step, target, sense_range,
                  search_comp, hold);
err = search_iv(&ib, &stat, &ic);
. . .
. . .
```

See Also

Search_iv
 Set_asearch
 Set_bsearch
 Set_sync (for search measurements)

Set_sync (for search measurements)

This function sets the synchronous port for performing synchronous search measurements.

Synopsis

```
#include <pcs/tis.h>

int set_sync(port, mode, offset,
             ratio, compliance, power_comp)
int port, mode;
double offset, ratio, compliance, power_comp;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, a pin number, or an unit address. <i>port address</i> 32701 to 32706, 32708, 32709, 32721 to 32732 ¹ <i>unit address</i> 32621 to 32633, 32635, 32637, 32639, . . . 32663 <i>pin number</i> 1 to 96 ²
<i>mode</i>	Specifies the polarity of the synchronization. You can choose from the following macros: POS_SYNC (= 0) NEG_SYNC (= 1)
<i>offset</i>	Specifies the offset added on the synchronous value relative to the primary value. Synchronous search port is SMU in VS mode [V]: -400 to 400 ³ Synchronous search port is VS [V]: -80 to 80 ³ Synchronous search port is SMU in IS mode [A]: -2 to 2 ³
<i>ratio, compliance, power_comp</i>	Used only for sweep synchronization.

¹ The port specified by a port address of 32705, 32706, 32708, 32709, 32721 to 32728, 32730, or 32731 must be connected to VS. If 32721 to 32732 is specified, the SWM must be equipped with a port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ Depending on search range set by Set_bsearch or Set_lsearch.

Description

This function, used in conjunction with Set_bsearch and Set_lsearch, sets the search parameters for the synchronous search port (SMU or VS).

The output mode of the synchronous search port is automatically set to the same output mode (VS or IS) as the search port of Set_bsearch or Set_lsearch function.

A port address, unit address, or pin number of any measurement pin connected to the port can be used to specify the synchronous search *port*.

The *mode* and *offset* arguments set the output of the synchronous search port as follows:

Mode = POS_SYNC:

Synchronous search port output = *offset* + primary search output

Mode = NEG_SYNC:

Synchronous search port output = *offset* – primary search output

The output range of the synchronous search port is the same range as the search port. When the synchronous search port is a VS, the output range of the search port must be set to 20 V or 40 V.

The following conditions must be satisfied:

For Set_bsearch:

$$S = \text{MAX}(|\text{min}|, |\text{max}|) + |\text{offset}|$$

For Set_lsearch:

$$S = \text{MAX}(|\text{start}|, |\text{stop}|) + |\text{offset}|$$

In VS mode

S [V]	Compliance [mA]	
	HP 41420A	HP 41421B
$0 < S \leq 2$	< 1000	< 100
$2 < S \leq 20$	< 700	< 100
$20 < S \leq 40$	< 350	< 50
$40 < S \leq 100$	< 125	< 20
$100 < S \leq 200$	< 50	—

Set_sync (for search measurements)

In IS mode

S [mA]	Compliance [V]	
	HP 41420A	HP 41421B
$0 < S \leq 20$	< 200	< 100
$20 < S \leq 50$	< 200	< 40
$50 < S \leq 100$	< 100	< 20
$100 < S \leq 125$	< 100	—
$125 < S \leq 350$	< 40	—
$350 < S \leq 700$	< 20	—
$700 < S \leq 1000$	< 14	—

Where *compliance* is the same as before this function executed.

Example

```
. . .
. . .
int drain, gate, substrate;
double search;
. . .
. . .
if (force_v(drain, -2.0, -5, 1.15e-5)
    == -1) error_rep();
if (set_bsearch(gate, drain, VS_NORM_ACC, 0.0, 5.0,
    -1e-5, -1e-6, 0, 0.1)
    == -1) error_rep();
if set_sync(substrate, POS_SYNC, 0.0, NULL, NULL, NULL)
    == -1) error_rep();
if (search_iv(&search, &stat, NULL) == -1) error_rep();
if (disable_port3(gate, drain, substrate)
    == -1) error_rep();
. . .
```

See Also

Set_asearch
Set_bsearch
Set_iv
Set_lsearch
Sweep_iv

Quasi-Pulsed Measurements

Introduction

This chapter provides the following functions for Quasi-pulsed measurements:

- Measure_bdv
- Measure_ileak
- Set_bdv
- Set_ileak

These functions let you set up and measure Quasi-pulsed measurements. The functions are listed in alphabetical order within this chapter.

For more information on quasi-pulsed measurements, see the *HP 4142B Operation Manual*.

Measure_bdv

This function triggers quasi-pulsed measurements to measure breakdown voltage, then returns the measured value.

Synopsis

```
#include <pcs/tis.h>

int measure_bdv(voltage, status, interval)
int *status, interval;
double *voltage;
```

Arguments

Item	Range Restrictions/Description
<i>voltage</i>	A pointer to a double where the measured voltage is stored. Any valid pointer
<i>status</i>	A pointer to an integer where the measurement status code is stored. The status is returned as one of the following codes: BDV_NORMAL (= 0) BDV_COMP_OTHER (= 1) BDV_CURRENT_LOW (= 2) BDV_OSC (= 3) BDV_OVERFLOW (= 4) BDV_TIMEOUT (= 6) BDV_TOO_SLOW (= 7)
<i>interval</i>	Specify the measurement interval with the following macros: BDV_INTVL_SHORT (= 0) BDV_INTVL_LONG (= 1)

Description

This function triggers the quasi-pulsed measurement to measure breakdown voltage. The conditions of the quasi-pulsed measurements are determined by the Set_bdv statement. The measured value and its measurement status are returned to the *voltage* and *status*, respectively.

If this function finishes successfully, a 0 is returned. Otherwise, a -1 is returned.

The following table shows the measurement status for the quasi-pulsed measurements of breakdown voltage:

Status Code	Condition
BDV_NORMAL	The quasi-pulsed measurement ended normally.
BDV_COMP_OTHER	Another unit reached compliance.
BDV_CURRENT_LOW	Breakdown voltage measurement failed because current did not reach the breakdown <i>current</i> .
BDV_OSC	This unit is oscillating.
BDV_OVERFLOW	Measurement overflow while monitoring output voltage or measuring leakage current.
BDV_TIMEOUT	The quasi-pulsed measurement did not reach breakdown <i>current</i> within the timeout.
BDV_TOO_SLOW	The monitored slew rate of the output voltage is too small.

Interval defines the monitoring duration of the output voltage. When the *interval* is set to BDV_INTVL_SHORT, the results of monitoring the BDV_INTVL_LONG mode 10 times are averaged and used to decide when the output current reaches the breakdown *current* value.

When the *interval* is set to BDV_INTVL_LONG, a slew rate of less than 1000 V/s (= 1 V/ms) causes an error (status = 7). When the *interval* is set to BDV_INTVL_SHORT, a slew rate of less than 100 V/s (= 0.1 V/ms) causes an error (status = 7).

The timeout for the quasi-pulsed measurement is set to 3 seconds for BDV_INTVL_SHORT and 12 seconds for BDV_INTVL_LONG.

For more information on quasi-pulsed measurements, see the *HP 4142B Operation Manual*.

Example

```
. . .
int st;
double v_meas;
. . .
if (measure_bdv(&v_meas,&st,BDV_INTV_SHORT)==-1)error_rep();
if (st > 5)
if (measure_bdv(&v_meas,&st,BDV_INTVL_LONG)==-1)error_rep();
. . .
```

See Also

Set_bdv

Measure_ileak

This function triggers the quasi-pulsed measurement to measure leakage current, then returns the measured value.

Synopsis

```
#include <pcs/tis.h>
```

```
int measure_ileak(port, current, status, interval)
int port, *status, interval;
double *current;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, an SMU unit address, or a pin number. Port address: 32701 to 32704, 32729 ¹ , 32732 ¹ Unit address: 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU) Pin number: 1 to 96 ²
<i>current</i>	A pointer to a double where the measured current is stored. Any valid pointer
<i>status</i>	A pointer to an integer where the measurement status code is stored. The status returned is one of the following codes: ILEAK_NORMAL (= 0) ILEAK_COMP_OTHER (= 1) ILEAK_COMP (= 2) ILEAK_OSC (= 3) ILEAK_OVERFLOW (= 4) ILEAK_TIMEOUT (= 6) ILEAK_TOO_SLOW(= 7)
<i>interval</i>	Specifies the measurement judge mode by the following macros: ILEAK_INTVL_SHORT (= 0) ILEAK_INTVL_LONG (= 1)

¹ These port addresses are used for the AUX ports of the port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

Description

This function triggers the quasi-pulsed measurement to measure leakage current. The conditions of the quasi-pulsed measurements are determined by the Set_ileak statement. The measured value and its measurement status are returned to *current* and *status*, respectively.

If this function finishes successfully, a 0 is returned. Otherwise, a -1 is returned.

A port address, SMU unit address, or pin number can be used to specify the measurement *port*.

The following table shows the measurement status for the quasi-pulsed measurements of leakage current:

Status Code	Condition
ILEAK_NORMAL	The quasi-pulsed measurement ended normally.
ILEAK_COMP_OTHER	Another unit reached compliance.
ILEAK_COMP	This unit reached compliance.
ILEAK_OSC	This unit is oscillating.
ILEAK_OVERFLOW	Measurement overflows either while monitoring the output voltage or measuring the leakage current.
ILEAK_TIMEOUT	The quasi-pulsed measurement did not reach output <i>voltage</i> within the timeout.
ILEAK_SLOW	The monitored slew rate of the output voltage is too small.

Interval defines the monitoring duration of the output voltage. When the *interval* is set to ILEAK_INTVL_LONG, the results of monitoring the ILEAK_INTVL_SHORT mode 10 times are averaged and used to decide when the output voltage reaches the output *voltage* value.

When the *interval* is set to ILEAK_INTVL_SHORT, a slew rate of less than 1000 V/s (= 1 V/ms) causes an error (status = 7). When the *interval* is set to ILEAK_INTVL_LONG, a slew rate of less than 100 V/s (= 0.1 V/ms) causes an error (status = 7).

The time out is set to 3 seconds for ILEAK_INTVL_SHORT and 12 seconds for ILEAK_INTVL_LONG.

For more information on the quasi-pulsed measurements, see the *HP 4142B Operation Manual*.

Measure_ileak

Example

```
. . .  
int st;  
double ileak;  
. . .  
if (measure_ileak(fnsmu(2), &ileak, &st, ILEAK_INTVL_SHORT) == -1) error_rep();  
if (st > 5)  
    if (measure_ileak(fnsmu(2), &ileak, &st, ILEAK_INTVL_LONG) == -1) error_rep();  
. . .
```

See Also

Set_ileak

Set_bdv

This function sets quasi-pulsed measurement parameters for breakdown voltage measurements.

Synopsis

```
#include <pcs/tis.h>
```

```
int set_bdv(port, range, start, stop, current, hold, delay)
int port;
double range, start, stop, current, hold, delay;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, an SMU unit address, or a pin number. Port address: 32701 to 32704, 32729 ¹ , 32732 ¹ Unit address: 32621 to 32628 (for HP 41421B 100 V/100 mA SMU); 32629 to 32632 (for HP 41420A 200 V/1 A SMU) Pin number: 1 to 96 ²
<i>range</i>	Specifies a measurement range. Numeric expression [V]: -200 to 200 ³ , 0.0 (Auto range)
<i>start</i>	Specifies the measurement ramp start voltage. Numeric expression [V]: -200 to 200 ³
<i>stop</i>	Specifies the measurement ramp stop voltage. Numeric expression [V]: -200 to 200 ³
<i>current</i>	Specifies the measurement current in which the breakdown voltage is measured. Numeric expression [A]: -1 to 1 ³
<i>hold</i>	Specifies a wait time taken before the output voltage starts from the <i>start</i> voltage. Numeric expression [s]: 0 to 655.35 Resolution: 0.01
<i>delay</i>	Specifies a hold time taken before the current value is measured at the SMU. Numeric expression [s]: 0 to 6.5535 Resolution: 0.0001

¹ These port addresses are used for the AUX ports of the port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ Actual *range*, *start*, *stop*, and *current* range restrictions depend on the SMU and SMU port used.

Set_bdv

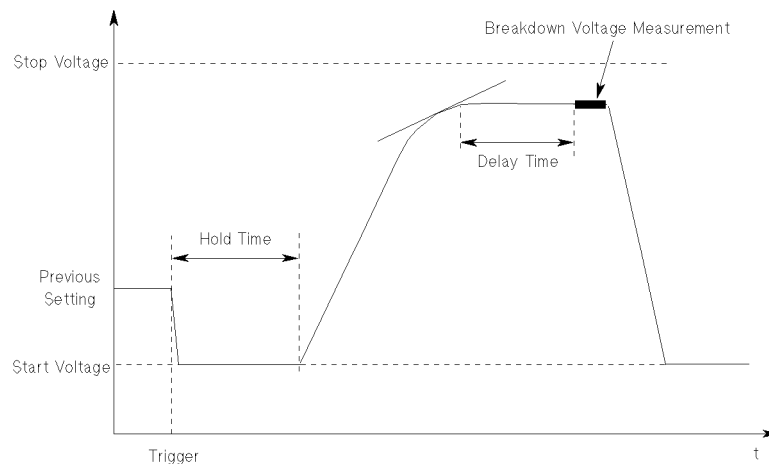
Description

This function sets the quasi-pulsed measurement conditions to measure breakdown voltage. To start quasi-pulsed measurements, the Measure_bdv function is used.

If this function finishes successfully, a 0 is returned. Otherwise, a -1 is returned.

A quasi-pulse is forced at a specified SMU and the forced voltage is monitored during a time interval. When the gradient of the voltage increase becomes less than the half of the initial gradient of the voltage increase, the SMU current reaches breakdown *current* (current compliance), and the voltage value is measured at the SMU. After the measurement, the forced voltage is reset to *start* voltage. The slew rate of the quasi-pulse is determined by capacitance on the DUT, cables, test fixtures, and so on.

The quasi-pulsed wave form for a breakdown voltage measurement is shown in the following illustration. These measurements are useful when the breakdown voltage range of the DUT is designed very wide and when the breakdown voltage must be measured at a low breakdown current so that the DUT is not damaged.



A port address, SMU unit address, or pin number can be used to specify the measurement *port*.

The breakdown voltage must be found within the *start* voltage and *stop* voltage. The *stop* voltage must not be the same value as the *start* value, and the difference between these two voltage values must be greater than or equal to 10 V.

Current is the breakdown current value when the breakdown voltage is measured. The breakdown current is current compliance of the SMU and must be low enough not to damage the DUT.

Hold takes a specified time before the output voltage starts from the *start* voltage. *Delay* is a specified hold time before the current value is measured at the SMU.

For more information on the quasi-pulsed measurements, see the *HP 4142B Operation Manual*.

8-8 Quasi-Pulsed Measurements

Example

```
int err;  
.  
.  
err=set_bdv(fnsmu(2),0.0,0.0,50.0,1e-5,0.5,0.01);  
.  
.  
.
```

See Also

Measure_bdv

Set_ileak

This function sets the quasi-pulsed measurement parameters for leakage current measurements.

Synopsis

```
#include <pcs/tis.h>
```

```
int set_ileak(port, range, voltage, compliance, start, hold, delay)
int port;
double range, voltage, compliance, start, hold, delay;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM port address, an SMU unit address, or a pin number. Port address: 32701 to 32704, 32729 ¹ , 32732 ¹ Unit address: 32621 to 32628 (for HP 41421B 100 V/100 mA SMU), 32629 to 32632 (for HP 41420A 200 V/1 A SMU) Pin number: 1 to 96 ²
<i>range</i>	Specifies a measurement range. Numeric expression [V]: –200 to 200 ³ , 0.0 for Auto range
<i>voltage</i>	Specifies the measurement voltage in which the leakage current is measured. Numeric expression [V]: –200 to 200 ³
<i>compliance</i>	Specifies the current or current compliance of the <i>port</i> . REMAIN macro or: Numeric expression [A]: –1 to 1 ³
<i>start</i>	Specifies the measurement ramp start voltage. REMAIN macro or: Numeric expression: [V]: –200 to 200 ³
<i>hold</i>	Specifies a hold time before the output voltage starts from the <i>start</i> voltage. Numeric expression [s]: 0 to 655.35 Resolution: 0.01
<i>delay</i>	Specifies a delay time before the current value is measured at the SMU. Numeric expression [s]: 0 to 6.5535 Resolution: 0.0001

¹ These port addresses are used for the AUX ports of the port expander.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

³ Actual output *range*, output *voltage*, *current* compliance, and *start* voltage range restrictions depend on the SMU and SMU port used.

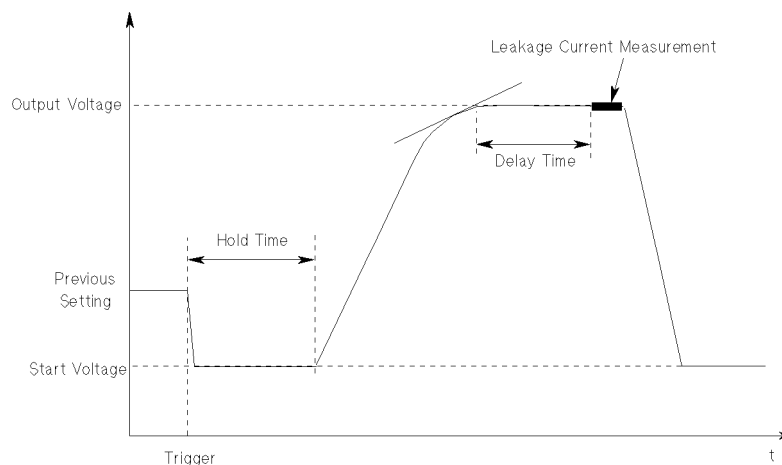
Description

This function sets the quasi-pulsed measurement conditions to measure leakage current. To start a quasi-pulsed measurement, a Measure_ileak function is used.

If this function finishes successfully, a 0 is returned. Otherwise, a -1 is returned.

A quasi-pulse to the output *voltage* value is forced at a specified SMU and the forced voltage is monitored during a time interval. When the gradient of the voltage increase becomes less than half of the initial gradient of the voltage increase, the current value is measured at the SMU. After the measurement, the forced voltage is reset to the *start* voltage. The slew rate of the quasi-pulse is determined by capacitance on the DUT, cables, test fixtures, and so on.

The quasi-pulsed wave form for a leakage current measurement is shown in the following illustration. These measurements are useful when a high voltage is forced to a DUT and low leakage current is measured.



A port address, SMU unit address, or pin number can be used to specify the measurement *port*.

Voltage, which corresponds to the stop voltage of the quasi-pulsed measurement, specifies the target voltage where the leakage current is measured.

Compliance limits the current flow through the DUT. *Start*, which corresponds to the start voltage of the quasi-pulsed measurements, specifies the start voltage of the quasi-pulse. You can use the REMAIN macro for *compliance* and *start*.

REMAIN specifies that the compliance or ramp voltage remains at the present setting unless the unit changes the voltage or current source mode by this function. For *start*, if the specified unit changed VS/IS mode by this function, REMAIN sets the start voltage to 0 V.

For compliance, if the specified unit changes voltage or current source status, specifying the REMAIN macro sets the current compliance to 100 μ A or sets the voltage compliance to 20 V, if the forcing current is less than or equal to 100 mA; otherwise the voltage compliance is set to 2 V.

Set_ileak

The *voltage* must not be the same value as the *start*, and the difference between these two voltages must be greater than or equal to 10 V.

Hold takes a specified time before the output voltage starts from the *start* voltage.

Delay takes a specified delay time before the current value is measured at the SMU.

For more information on quasi-pulsed measurements, see the *HP 4142B Operation Manual*.

Example

```
. . . .  
int err;  
. . . .  
err=set_ileak(fnsmu(1),100.0,60.0,1E-5,10.0,0.5,0.01);  
. . . .
```

See Also

Measure_ileak

DC Source Output Relay Control

Introduction

The C Library provides the following functions to control the HP 4142B dc source/monitor output relays:

- Ch_sw_off
- Ch_sw_off_all
- Ch_sw_on
- Ch_sw_on_all

These functions set the output switch of the DCS output unit(s) to on or off and sets or clears the mask flag of the output unit. One unit at a time may be set to on or off, or all the units may be set to on or off.

Only one unit is controlled by one function, so repeat the function to control several units. Use Ch_sw_on_all or Ch_sw_off_all to control all the units at once.

The functions are arranged alphabetically in this chapter.

Ch_sw_off

This function sets the output switch of a specified DCS output unit to off (disables output).

Synopsis

```
#include <pcs/tis.h>

int ch_sw_off(port)
int port;
```

Argument

Item	Range Restrictions/Description
<i>port</i>	An SWM number, pin number, or a unit address.
<i>port address</i>	32701 to 32704, 32729, 32732 (SMU), 32705, 32706, 32708, 32709 32721 to 32728, 32730, 32731 ¹
<i>unit address</i>	32621 to 32628 (HP 41421B SMU) 32629 to 32632 (HP 41420A SMU) 32633, 32635, 32637 . . . 32663 (for HP 41424A VSs) ²
<i>pin number</i>	1 to 96 ³

- ¹ If you specify 32729, 32732, 32721 to 32728, 32730, or 32731, your SWM must be equipped with a port expander.
- ² The VMs of the VS/VMU do not have output switches.
- ³ 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

Description

This function sets the output switch of the specified DCS output unit to off, and sets the mask flag of the output unit. A port address, a unit address, or a pin number can be used to specify the output unit *port*. Mask flags are cleared by the Ch_sw_on function only.

Only one relay can be controlled by one call. Repeat the Ch_sw_off function, or use the Ch_sw_off_all function if two or more relays need to be controlled.

Note

This function cannot be used if an output unit is set to the High Voltage state (more than 42 V).



When this function executes, the specified output unit(s) are set as follows:

	SMU	VS
output switch	OFF	OFF
output mode	voltage	—
output voltage	0 V (20 V range)	0 V (20 V range)
compliance/limiter	100 μ A compliance (100 μ A range)	100 mA limiter
SMU filter	ON	—

The following TIS functions also control the output switch of the DCS output units, as described in the table below:

Function	Description
Init_system	Sets all output switches to off and clears all mask flags.
Force_v, Force_i, Set_iv, Set_piv, Set_pbias, Measure_p, Set_sync, Set_asearch, Set_bsearch, Set_lsearch	Sets all output switches to on (except the masked units).

Example

```
. . .
int err;
. . .
err = ch_sw_off(32621);
. . .
```

See Also

Ch_sw_off_all
Ch_sw_on
Ch_sw_on_all

Ch_sw_off_all

This function sets the output switches of the DCS output unit(s) to off (disables output). All the dc source output relay control mask bits are set at one time.

Synopsis

```
#include <pcs/tis.h>

int ch_sw_off_all()
```

Description

This function sets the output switches of all the DCS output units to off, and sets the mask flag of the output units.

When this function executes, the specified output units)are set as follows:

	SMU	VS
output switch	OFF	OFF
output mode	voltage	—
output voltage	0 V (20 V range)	0 V (20 V range)
compliance/limiter	100 μ A compliance (100 μ A range)	100 mA limiter
SMU filter	ON	—

The following TIS functions also control the output switch of the DCS output units, as described in the table below:

Function	Description
Init_system	Sets all output switches to off and clears all mask flags.
Force_v, Force_i, Set_iv, Set_piv, Set_pbias, Measure_p, Set_sync, Set_asearch, Set_bsearch, Set_lsearch	Sets all output switches to on (except the masked units).

Example

```
. . .
int err;
. . .
err = ch_sw_off_all();
. . .
```

See Also

Ch_sw_off

Ch_sw_on

Ch_sw_on_all

Ch_sw_on

This function sets the output switch of the specified DCS output unit to on (enables output).

Synopsis

```
#include <pcs/tis.h>
```

```
int ch_sw_on(port)
int port;
```

Argument

Item	Range Restrictions/Description
<i>port</i>	An SWM port number, pin number, or unit address.
<i>port address</i>	32701 to 32704 (SMU), 32705, 32706, 32708, 32709 32721 to 32728, 32730, 32731 ¹
<i>unit address</i>	32621 to 32628 (HP 41421B SMU) 32629 to 32632 (HP 41420A SMU) 32633, 32635, 32637 . . . 32663 (HP 41424A VSs) ²
<i>pin number</i>	1 to 96 ³

¹ If you specify 32729, 32732, 32721 to 32728, 32730, or 32731, your SWM must be equipped with a port expander.

² The VMs of the VS/VMU do not have output switches.

³ 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the desired port.

Description

This function sets the output switch of the specified DCS output unit to on, and clears the mask flag of the output unit. A port address, a unit address, or a pin number can be used to specify the *port* of the output unit. Mask flags are set by the Ch_sw_off function only.

Only one relay can be controlled by one call, so repeat the Ch_sw_on function or use Ch_sw_on_all if two or more relays need to be controlled.

Note

This function cannot be used if an output unit is set to the High Voltage state (more than 42 V).



When this function executes, the specified output units are set as follows:

	SMU	VS
output switch	OFF	OFF
output mode	voltage	—
output voltage	0 V (20 V range)	0 V (20 V range)
compliance/limiter	100 μ A compliance (100 μ A range)	100 mA limiter
SMU filter	ON	—

The following TIS functions also control the output switch of the DCS output units as described below:

Function	Description
Init_system	Sets all output switches to off and clears all mask flags.
Force_v, Force_i, Set_iv, Set_piv, Set_pbias, Measure_p, Set_sync, Set_asearch, Set_bsearch, Set_lsearch	Sets all output switches to on.

Example

```

. . .
int err;
. . .
err = ch_sw_on(32629)
. . .

```

See Also

Ch_sw_off
 Ch_sw_off_all
 Ch_sw_on_all

Ch_sw_on_all

This function sets all the output switches of the DCS output unit(s) to on (enables output). All dc source output relay control mask bits are cleared at once.

Synopsis

```
#include <pcs/tis.h>

int ch_sw_on_all()
```

Description

This function sets the output switches of the DCS output units to on, and clears all the mask flags of the output unit.

When this function executes, the specified output units are set as follows:

	SMU	VS
output switch	OFF	OFF
output mode	voltage	—
output voltage	0 V (20 V range)	0 V (20 V range)
compliance/limiter	100 μ A compliance (100 μ A range)	100 mA limiter
SMU filter	ON	—

The following TIS functions also control the output switch of the DCS output units as described below:

Function	Description
Init_system	Sets all output switches to off and clears all mask flags.
Force_v, Force_i, Set_iv, Set_piv, Set_pbias, Measure_p, Set_sync, Set_asearch, Set_bsearch, Set_lsearch	Sets all output switches to on (ignores mask flag setting).

Example

```
. . .
int err;
. . .
err = ch_sw_on_all();
. . .
```

See Also

Ch_sw_off
Ch_sw_off_all
Ch_sw_on

Capacitance Measurements

Introduction

The C Library provides the following functions for capacitance measurements:

- Measure_cmu
- Measure_cmu84
- Open_corr84
- Set_cmu
- Set_cmu84
- Set_ct
- Set_cv
- Set_cv84
- Set_freq
- Short_corr84
- Sweep_ct
- Sweep_cv
- Sweep_cv84

These functions set the parameters for capacitance and conductance measurements, and perform the measurements. The functions are arranged alphabetically in this chapter.

Measure_cmu

This function uses the CMU to measure capacitance and conductance, and returns the error corrected values.

Synopsis

```
#include <pcs/tis.h>
```

```
int measure_cmu(capacitance, conductance, range)
double *capacitance, *conductance, range;
```

Arguments

Item	Range Restrictions/Description
<i>capacitance</i>	A pointer to a double where the measured capacitance is stored. Any valid pointer
<i>conductance</i>	A pointer to a double where the measured conductance is stored. Specify NULL to discard measured value OR: Any valid pointer
<i>range</i>	Specifies the measurement range. 0.0 for Auto range OR: > 0.0 and $\leq 1.0\text{E}-9$ for capacitance range. Between $1.0\text{E}-9$ and $1.0\text{E}-2$ for conductance range.

Description

This function returns the error-corrected measurement results to the specified *capacitance* name and *conductance* name. Measure_cmu provides automatic correction for the residuals of the signal cables that connect the CMU to the SWM, and the residuals of the pin boards. Measurement values include the residual impedance that exists between the SWM and the device under test. The measurement *range* argument sets the C·G range of the CMU. Refer to the following table.

Measurement Range Value	C Range (Maximum) ¹	G Range (Maximum) ¹
$0 < \text{value} \leq 1\text{E}-11$ $1\text{E}-9 < \text{value} \leq 1\text{E}-4$	10 pF (19.00 pF)	100 μS (120.0 μS)
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$ $1\text{E}-4 < \text{value} \leq 1\text{E}-3$	100 pF (190.0 pF)	1 mS (1.200 mS)
$1\text{E}-10 < \text{value} \leq 1\text{E}-9$ $1\text{E}-3 < \text{value} \leq 1\text{E}-2$	1000 pF (1900 pF)	10 mS (12.00 mS)
0.0	Auto range ²	

¹ Maximum value that can be displayed on the front panel of the CMU.

10-2 Capacitance Measurements

² The optimum C·G Range for the capacitance measurement value is automatically selected. It is not necessary to define the exact range value for the CMU; an arbitrary value is accepted as long as it is within the allowable range.

The measurement resolution of each range depends on the integration time setting. Refer to the Set_cmu function. If a measurement overflow occurs, 9.99999E+9 is returned to the *capacitance* name, and 9.9999E+9 is returned to the *conductance* name.

In Offline Mode:

This function returns simulated measurement values to the *capacitance* and *conductance* arguments. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

Returned values are determined by the measurement *range* value as follows:

Measurement Range Value	Capacitance	Conductance
0	10 pF	100 μ S
$0 < \text{value} \leq 1\text{E}-11$	<i>range</i> value	100 μ S
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$	<i>range</i> value	1 mS
$1\text{E}-10 < \text{value} $	<i>range</i> value	10 mS

File Data Simulation mode:

This function reads <c_meas> and <g_meas> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Measure_cmu  <c_meas>  <g_meas>
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . .
int err;
double capacitance, conductance;
. . .
err = force_v(fncmh(), 5.0, 20.0, 1E-3);
err = measure_cmu(&capacitance, &conductance, 0.0);
. . .
. . .
```

See Also

Port_status

Measure_cmu84

This function uses a CMU84 to measure capacitance and conductance, and returns the error corrected values.

Synopsis

```
#include <pcs/tis.h>
```

```
int measure_cmu84(capacitance, conductance, range)
double *capacitance, *conductance, range;
```

Arguments

Item	Range Restrictions/Description
<i>capacitance</i>	A pointer to a double where the measured capacitance is stored. Any valid pointer
<i>conductance</i>	A pointer to a double where the measured conductance is stored. Specify NULL to discard measured value OR: Any valid pointer
<i>range</i>	Specifies the measurement range in a capacitance range. The Measure_cmu84 functions translates the capacitance range to the impedance range, which the HP 4284A accepts. Specify 0.0 for Auto range mode OR: Numeric expression [F]: real value

Description

This function performs a capacitance measurement with the CMU84 set to Cp-G mode and returns the error compensated capacitance and conductance values to *capacitance* and *conductance*, respectively.

The measurement *range* sets the capacitance measurement range (C range) of the CMU84. The conductance measurement range (G range) is automatically set to measurement *range* $\times 1.0 \times 10^7$. If 0.0 is specified as the argument value, the CMU84 uses the Auto range mode. If a value other than 0.0 is specified, the following rule applies for the measurement range of the CMU84.

You can specify a capacitance range as this argument, but the CMU84 performs the impedance measurement according to the impedance range. The relationship between the capacitance and impedance range is shown in the following equation:

$$\text{impedance range} = \frac{1}{2\pi f |\text{measurement range}|}$$

Where, f is the measurement frequency.

The specified measurement and impedance ranges are shown below:

10-4 Capacitance Measurements

measurement range restrictions [F]	measurement frequencies			
	1 kHz	10 kHz	100 kHz	1 MHz
$0 \leq \text{value} \leq 1.592\text{E}-12$	100 k Ω	100 k Ω	100 k Ω	100 k Ω
$1.592\text{E}-12 < \text{value} \leq 5.305\text{E}-12$	100 k Ω	100 k Ω	100 k Ω	30 k Ω
$5.305\text{E}-12 < \text{value} \leq 1.592\text{E}-11$	100 k Ω	100 k Ω	100 k Ω	10 k Ω
$1.592\text{E}-11 < \text{value} \leq 5.305\text{E}-11$	100 k Ω	100 k Ω	30 k Ω	3 k Ω
$5.305\text{E}-11 < \text{value} \leq 1.592\text{E}-10$	100 k Ω	100 k Ω	10 k Ω	1 k Ω
$1.592\text{E}-10 < \text{value} \leq 5.305\text{E}-10$	100 k Ω	30 k Ω	3 k Ω	300 Ω
$5.305\text{E}-10 < \text{value} \leq 1.592\text{E}-9$	100 k Ω	10 k Ω	1 k Ω	100 Ω
$1.592\text{E}-9 < \text{value} \leq 5.305\text{E}-9$	30 k Ω	30 k Ω	300 Ω	100 Ω
$5.305\text{E}-9 < \text{value} \leq 1.592\text{E}-8$	10 k Ω	1 k Ω	100 Ω	100 Ω
$1.592\text{E}-8 < \text{value} \leq 5.305\text{E}-8$	3 k Ω	300 Ω	100 Ω	10 Ω
$5.305\text{E}-8 < \text{value} \leq 1.592\text{E}-7$	1 k Ω	100 Ω	100 Ω	10 Ω
$1.592\text{E}-7 < \text{value} \leq 5.305\text{E}-7$	300 Ω	100 Ω	10 Ω	1 Ω
$5.305\text{E}-7 < \text{value} \leq 1.592\text{E}-6$	100 Ω	100 Ω	10 Ω	1 Ω
$1.592\text{E}-6 < \text{value} \leq 1.592\text{E}-5$	100 Ω	10 Ω	1 Ω	1 Ω
$1.592\text{E}-5 < \text{value} \leq 1.592\text{E}-4$	10 Ω	1 Ω	1 Ω	1 Ω
$1.592\text{E}-4 < \text{value} $	1 Ω	1 Ω	1 Ω	1 Ω

Refer to the *HP 4284A Precision LCR Meter Operation Manual* for more information on the impedance range.

It is not necessary to define the exact range value for the CMU84; an arbitrary value is accepted as long as it is within the allowable range.

Measure_cmu84

In Offline Mode:

In offline mode execution, this function returns dummy data. Refer to the following table.

measurement range	dummy capacitance value	dummy conductance value
AUTO	previous C range value	previous G range value
other than AUTO	C range value	G range value

Example

```
. . . .  
int err;  
double capacitance, conductance;  
. . . .  
err = force_v(fncmh(), 5.0, 20.0, 1E-3);  
err = measure_cmu84(&capacitance, &conductance, 0.0);  
. . . .
```

See Also

Port_status

Open_corr84

This function performs the OPEN correction of the CMU84.

Synopsis

```
#include <pcs/tis.h>

int open_corr84()
```

Description

This function triggers the OPEN compensation function in the CMU84, and compensation data is stored in the internal non-volatile memory of the CMU84. Refer to the *HP 4284A Operation Manual* for more information.

Example

```
. . .
open_corr84();
. . .
```

See Also

Short_corr84

Set_cmu

This function sets the measurement integration time and test signal level of the CMU.

Synopsis

```
#include <pcs/tis.h>

int set_cmu(integ_time, sig_level)
int integ_time, sig_level;
```

Arguments

Item	Range Restrictions/Description
<i>integ_time</i>	Specifies the measurement integration time of the CMU. Choose from the following macros: INTEG_SHORT (= 1) INTEG_MEDIUM (= 2) INTEG_LONG (= 3)
<i>test sig_level</i>	Specifies the test signal level. You can choose from two macros: SIG_LEVEL_LOW (= 1): 10 mVrms SIG_LEVEL_HIGH (= 2): 30 mVrms

Description

Use *integ_time* to choose the measurement integration time of the CMU. You have three macros to choose from: INTEG_SHORT for 1 ms integration time, INTEG_MEDIUM for 10 ms integration time, and INTEG_LONG for an average of 10 MEDIUM integration time measurements.

The test signal level of the CMU is set to 10 mVrms when you select SIG_LEVEL_LOW, and 30 mVrms when you select SIG_LEVEL_HIGH. Measurement resolutions for the set integration times are listed as follows:

Range	SHORT	MEDIUM	LONG
10 pF	10 pF	1 fF	
100 pF	0.1 pF	10 fF	
1000 pF	1 pF	0.1 pF	

Integration time and test signal level are initially set to MEDIUM and 30 mVrms, respectively.

Example

```
. . .  
int err;  
. . .  
err = set_cmu(INTEG_LONG, SIG_LEVEL_HIGH);  
. . .
```

See Also

Measure_cmu

Set_cmu84

This function sets the measurement integration time and test signal level of the CMU84.

Synopsis

```
#include <pcs/tis.h>

int set_cmu84(integ_time, sig_level)
int integ_time;
double sig_level;
```

Arguments

Item	Range Restrictions/Description
<i>integ_time</i>	You can choose from the following macros for the integration time: INTEG_SHORT (= 1) INTEG_MEDIUM (= 2) INTEG_LONG (= 3)
<i>sig_level</i>	The range allowed for the HP 4284A measurement signal level depends on the HP 4284A product option that specifies the HP 4284A signal level extension. REMAIN macro OR: Numeric expression [V] 0 to 2.0 (standard) 0 to 20.0 (option 531)

Description

Use *integ_time* to choose the measurement integration time of the CMU84. You have three macros to choose from: INTEG_SHORT, INTEG_MEDIUM, and INTEG_LONG. The following table shows typical measurement times.

	1 kHz	10 kHz	100 kHz	1 MHz
SHORT	40 ms	30 ms	30 ms	30 ms
MEDIUM	190 ms	180 ms	180 ms	180 ms
LONG	830 ms	820 ms	820 ms	820 ms

The test *sig_level* argument defines the test signal voltage. The test signal level consideration is as follows:

- Option 531 (HP 4284A option 001) *not* installed

Specified Signal level [V]	Actual Signal Level	Resolution
$\text{specification} \leq -0.005$	error	—
$-0.005 < \text{specification} < 0.005$	0.000 V	—
$0.005 \leq \text{specification} \leq 0.200$	specification	0.001 V
$0.200 < \text{specification} \leq 2.00$	specification	0.01 V
$2.00 < \text{specification} < 2.01$	2.00 V	—
$2.01 \leq \text{specification}$	error	—

- Option 531 (HP 4284A option 001) installed

Specified Signal level [V]	Actual Signal Level	Resolution
$\text{specification} \leq -0.005$	error	—
$-0.005 < \text{specification} < 0.005$	0.000 V	—
$0.005 \leq \text{specification} \leq 0.200$	specification	0.001 V
$0.200 < \text{specification} \leq 2.00$	specification	0.01 V
$2.00 < \text{specification} \leq 20.0$	specification	0.1 V
$20.0 < \text{specification} < 20.1$	20.0 V	—
$20.1 \leq \text{specification}$	error	—

Example

```

. . .
int err;
. . .
err = set_cmu84(INTEG_LONG, 1.0);
. . .
. . .

```

See Also

Measure_cmu84

Set_ct

This function sets the sweep parameters for C·G-t measurements.

Synopsis

```
#include <pcs/tis.h>

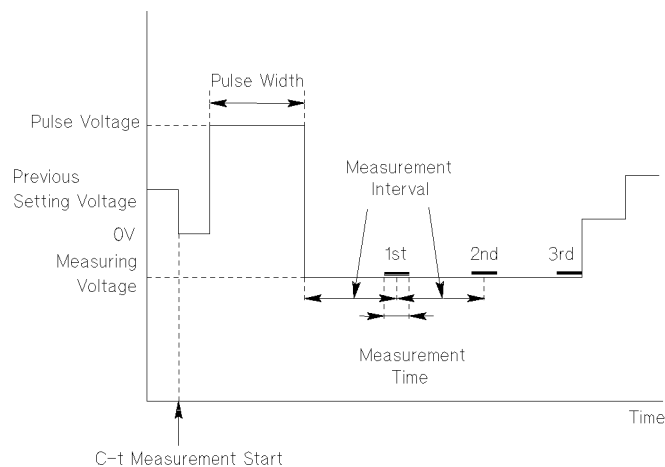
int set_ct(pulse_voltage, meas_voltage, number, width, interval)
int number;
double pulse_voltage, meas_voltage, width, interval;
```

Arguments

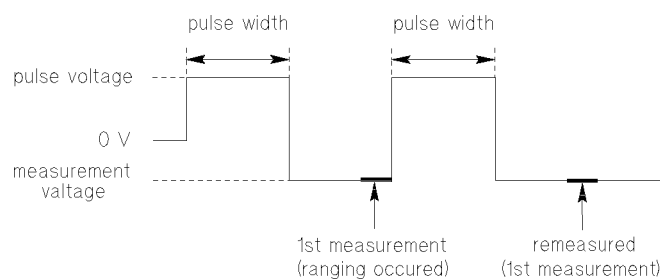
Item	Range Restrictions/Description
<i>pulse_voltage</i>	Specifies the peak voltage of the stimulus pulse. Numeric expression [V]: −100 to 100
<i>meas_voltage</i>	Specifies the bias voltage of the actual measurements, which are performed after the stimulus pulse. Numeric expression [V]: −100 to 100
<i>number</i>	Specifies the number of measurements. 2 to 1001
<i>width</i>	Specifies the stimulus pulse width. Numeric expression [s]: 0.01 to 32
<i>interval</i>	Specifies the measurement interval (the time of the measurement itself). Numeric expression [s]: 0.01 to 32

Description

When Sweep_ct executes, *pulse_voltage*, *meas_voltage*, *width*, and *interval* determine C·G-t measurement timing, as shown in the following figure:



If the C-G range is set to Auto range mode and the first measurement results in an overflow, the CMU automatically up-ranges and reapplies the pulse voltage.



The selected dc bias range depends on the *pulse_voltage* or *meas_voltage* value, whichever is larger.

DC Bias Range	Maximum Output Voltage	Resolution
1 V	± 1.999 V	1 mV
10 V	± 19.99 V	10 mV
100 V	± 100.0 V	100 mV

Set_ct

The measurement interval specified depends on the integration time, which is set with the *integ_time* of the Set_cmu function. If you select INTEG_LONG, integration time automatically reverts to INTEG_MEDIUM during a sweep. Pulse width and measurement interval resolution depend on their specified values, as shown in the following table.

Item	Range	Resolution				
		0.01	0.065	1	10	32
Measurement Interval (When integration time is SHORT)	0.06 s to 32 s	10 us	500 us	1 ms	10 ms	
Measurement Interval (When integration time is MEDIUM)	0.1 s to 32 s		500 us	1 ms	10 ms	
Pulse Width	0.01 s to 32 s	10 us	500 us	1 ms	10 ms	

Example

```
. . .
int err;
int number;
double v_pulse, v_meas, number, p_width, interval;
double range, capa[21], cond[21], time[21];
. . .
err = set_ct(v_pulse, v_meas, number, p_width, interval);
err = sweep_ct(range, capa, cond, time);
. . .
. . .
```

See Also

- Set_cv
- Sweep_ct
- Sweep_cv

Set_cv

This function sets the sweep parameters for C-G-V measurements.

Synopsis

```
#include <pcs/tis.h>

int set_cv(start, stop, number, hold, delay)
int number;
double start, stop, hold, delay;
```

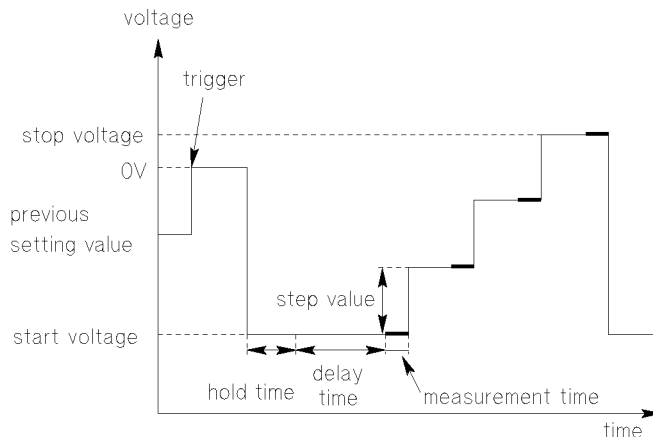
Arguments

Item	Range Restrictions/Description
<i>start, stop</i>	Specifies start and stop values of the CMU dc bias voltage. Numeric expression [V]: -100 to 100
<i>number</i>	Specifies the number of sweep steps. 2 to 1001
<i>hold</i>	Specifies a wait time taken after the CMH port voltage is set to <i>start</i> . If a non-zero <i>delay</i> is set, the <i>delay</i> time is added to this value prior to the measurement at the first sweep step. Numeric expression [s]: 0.003 to 650
<i>delay</i>	Specifies an amount of time after each measurement step and before the actual measurement. This delay is also used in conjunction with the hold time during the initial measurement. Numeric expression [s]: 0.003 to 650

Description

When Sweep_cv is executed, *start*, *stop*, *hold*, and *delay* determine the sweep measurement conditions, as shown in the following illustration:

Set_cv



The selected dc bias range depends on the *start* or *stop* voltage, whichever is larger. The step voltage— $(stop - start)/(number - 1)$ —may be rounded off, depending on the resolution of the set bias range.

DC Bias Range	Maximum Output Voltage	Resolution
1 V	± 1.999 V	1 mV
10 V	± 19.99 V	10 mV
100 V	± 100.0 V	100 mV

Because of the rounding-off, the actual *stop* can be slightly different than the specified value, and the sweep measurement can finish before the number of measurements reach the specified number of steps.

Hold time and delay time resolutions depend on their specified value, as listed in the following table:

Hold Time or Delay Time [s]	Resolution
0.003 to 0.065	1 ms
0.07 to 99.99	10 ms
100.0 to 650.0	100 ms

Example

```
. . .  
int err;  
int number;  
double v_start, v_stop, hold, delay;  
double range, capa[21], cond[21], bias[21];  
. . .  
err = set_cv(v_start, v_stop, number, hold, delay);  
err = sweep_cv(range, capa, cond, bias);  
. . .
```

See Also

Set_ct
Sweep_ct
Sweep_cv

Set_cv84

This function sets the sweep parameters for C·G·V measurements on the CMU84. Note that Option 531 (HP 4284A option 001) must be installed in order to perform the C·G·V measurements on the CMU84.

Synopsis

```
#include <pcs/tis.h>

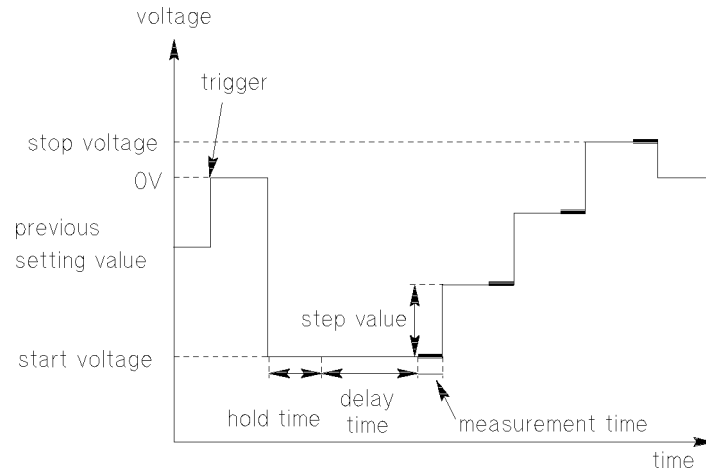
int set_cv84(start, stop, number, hold, delay)
int number;
double start, stop, hold, delay;
```

Arguments

Item	Range Restrictions/Description
<i>start, stop</i>	Specifies the start and stop values of the CMU84 dc bias voltage. Numeric expression [V]: -40.0 to 40.0
<i>number</i>	Specifies the number of sweep steps. 2 to 1001
<i>hold</i>	Specifies a wait time taken after the CMH84 port voltage is set to <i>start</i> . If a non-zero <i>delay</i> is set, the <i>delay</i> is added to this value prior to the measurement at the first sweep step. Numeric expression [s]: 0 to 650 Resolution = 1 ms
<i>delay</i>	The wait time specified for this argument is taken behind each measurement step and after the bias voltage change. Numeric expression [s]: 0 to 650 Resolution = 1 ms

Description

The C·G·V measurement concept is shown in the following figure.



The i -th output bias voltage is calculated from *start voltage*, *stop voltage*, and *number of steps* by using the following equation:

$$i\text{th output bias voltage} = \text{start voltage} + \frac{\text{stop voltage} - \text{start voltage}}{\text{number of steps} - 1} \times (i - 1)$$

$$i = 1, 2, \dots, \text{number of steps}$$

The output bias voltage is rounded off in the same manner as the Force_v function. Refer to the CMU84 bias voltage explanation of the Force_v function.

If the *start* voltage value is the same as the *stop* voltage value, an error occurs.

Example

```
. . .
int err;
int number;
double v_start, v_stop, hold, delay;
double range, capa[21], cond[21], bias[21];
. . .
err = set_cv84(v_start, v_stop, number, hold, delay);
err = sweep_cv84(range, capa, cond, bias);
. . .
. . .
```

See Also

Sweep_cv84

Set_freq

This function sets the measurement frequency of the CMU84.

Synopsis

```
#include <pcs/tis.h>

int set_freq(port, frequency)
int port;
double frequency;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies an SWM CML or CMH port address, a pin number that is connected to a CMH or CML port, or the unit address of a CMH84 or CML84. <i>port address</i> 32708, 32709, 32727, 32728, 32730, 32731 ¹ <i>unit address</i> 32676, 32677 <i>pin number</i> 1 to 96 ²
<i>frequency</i>	Specifies the measurement frequency. The value is rounded to the nearest measurement signal frequency that the HP 4284A provides. Numeric expression [Hz]: 1E3, 1E4, 1E5, and 1E6

¹ Must be the CMH and CML terminals of the CMU84 connected to AUX3 (CMH) and AUX4 (CML) ports, respectively.

² 1 to 48 for HP 4085B 48-pin SWM systems. Must be the number of a measurement pin connected to the CMH port.

Description

A value returned by Fncmh, Fncml, Fncmu84, Fnaux(31), Fnaux(32), Fnaux(41), or Fnaux(42) is suitable for the *port* argument. Note that there is no difference between specifying CMH84 and CML84.

The measurement *frequency* can be set to 1 kHz, 10 kHz, 100 kHz, or 1 MHz with this function.

Note



Frequencies between 20 Hz through 1 MHz are available for *frequency* argument and the measurement is executed without an error message. However, Hewlett-Packard guarantees the accuracy of measurement data taken with only the following frequencies: 1 kHz, 10 kHz, 100 kHz, and 1 MHz.

Example

```
. . .
int err;
. . .
err = set_freq(fncmh, 1.0e6);
. . .
```

See Also

Measure_cmu84

Set_cmu84

Short_corr84

This function performs SHORT compensation on the CMU84.

Synopsis

```
#include <pcs/tis.h>

int short_corr84()
```

Description

This function triggers the SHORT compensation function in the CMU84, and compensation data is stored in the internal non-volatile memory of the CMU84. Refer to the *HP 4284A Operation Manual* for more information.

Example

```
. . .
Short_corr84();
. . .
```

See Also

Open_corr84

Sweep_ct

This function performs C-G-t measurements on the CMU in accordance with the conditions established by the Set_ct function, and returns corrected values and measurement points.

Synopsis

```
#include <pcs/tis.h>

int sweep_ct(range, capacitance, conductance, time)
double range, capacitance[], conductance[], time[];
```

Arguments

Item	Range Restrictions/Description
<i>range</i>	Specifies the measurement range. 0.0 for Auto range mode or numeric expression [F] or [S] ¹ : 0 to 0.01
<i>capacitance</i>	A pointer to a double array where the measured capacitance values are stored. Any valid pointer [F]
<i>conductance</i>	A pointer to a double array where the measured conductance values are stored. Specify NULL if conductance results not needed OR: Any valid pointer [S]
<i>time</i>	A pointer to a double array where the time stamp of each measurement is stored. Specify NULL if time results not needed or any valid pointer [s].

¹ If $0 < |\text{specified value}| \leq 1\text{E}-9$, the unit is farad F, and if $1\text{E}-9 < |\text{specified value}| \leq -2$, the unit is siemens S.

Description

This function returns capacitance and conductance measurement values and the times at which the measurements were made to the *capacitance*, *conductance*, and *time* arrays, respectively.

Sweep_ct compensates for the effects of the cables that interconnect the CMU, SWM, and the pin boards. However, the measurement value includes the stray capacitance between the pin boards and the test device. First measure the stray capacitance without the DUT connected, then subtract the stray capacitance value from the measured capacitance value.

The measurement resolution depends on the *integ_time* set in the Set_cmu function. Refer to the Set_cmu function for details.

The size of the *capacitance*, *conductance*, and *time* arrays should be larger than the *number* specified in the Set_ct function, so that -9999999.99999 is returned to the remaining array elements. If smaller, the measurement data that exceed the array size are written to an invalid memory location. This may cause the loss of some other variable content, a segment violation error, or a bus error.

Sweep_ct

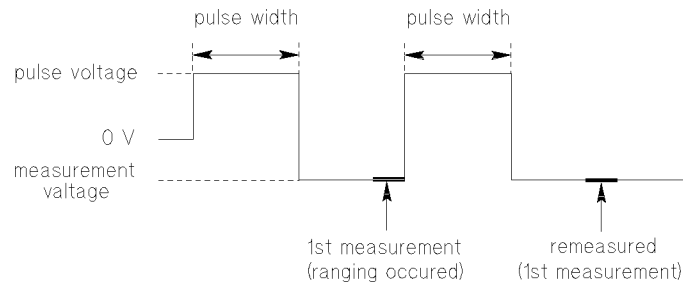
The C·G measurement range is determined by the *range* value as follows:

Measurement Range Value	C Range (Maximum ¹)	G Range (Maximum ¹)
$0 < \text{value} \leq 1\text{E}-11$ $1\text{E}-9 < \text{value} \leq 1\text{E}-4$	10 pF (19.00 pF)	100 μS (120 μS)
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$ $1\text{E}-4 < \text{value} \leq 1\text{E}-3$	100 pF (190.0 pF)	1 mS (1.2 mS)
$1\text{E}-10 < \text{value} \leq 1\text{E}-9$ $1\text{E}-3 < \text{value} \leq 1\text{E}-2$	1000 pF (1900 pF)	10 mS (12 mS)
0	Auto range	

¹ The maximum value displayed on the front panel of the HP 4280A.

It is not necessary to define an exact range value for the HP 4280A; an arbitrary value is acceptable as long as it is within the allowable range.

If the *range* is set to Auto range, and the first C·G-t measurement results in an overflow, the CMU automatically ranges up and reapplies the pulse voltage. However, for subsequent C·G-t measurements, the C·G range does not change, even if the measurement value exceeds the set range.



If overflow occurs during a measurement, twice the value of the set range is returned. For example, if a measurement is performed at the 1000 pF/10 mS range and overflow occurs, $2\text{E}-9$ and $2\text{E}-2$ is returned to *capacitance* and *conductance*, respectively.

The time after changing the measurement voltage to the *n*-th actual measurement is returned to the *time* array. The expected return data is as follows:

$$\text{Array}(i - 1) = \text{measurement interval} \times i$$

$$i = 1, 2, \dots, \text{number of readings}$$

Where the *interval* time and *number* of readings is set by Set_ct statement.

In Offline Mode:

This function returns simulated measurement values to the *capacitance*, *conductance*, and *time* arrays. Type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns a sequence of linear values to the *capacitance*, *conductance*, and *time* arrays. Linear values returned to the *time* array are determined by the *number* of readings and measurement *interval* arguments of the Set_ct function. Linear values returned to the *capacitance* and *conductance* arrays range from 0 to the value determined by the *range* as described in the following table.

Measurement Range Value	Capacitance	Conductance
0	10 pF	100 μ S
$0 < \text{value} \leq 1\text{E}-11$	range value	100 μ S
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$	range value	1 mS
$1\text{E}-10 < \text{value}$	range value	10 mS

File Data Simulation mode:

This function reads <c_meas>, <g_meas>, and <time> from the simulation file as the capacitance measurement, conductance measurement, and measurement point values, respectively. The data simulation file format is as follows:

```
Sweep_ct
[
<c_meas>  <g_meas>  <time>
<c_meas>  <g_meas>  <time>
. . . . .
<c_meas>  <g_meas>  <time>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Sweep_ct

Example

```
. . .  
int err;  
int number;  
double v_pulse, v_meas, number, p_width, interval;  
double range, capa[21], cond[21], time[21];  
. . .  
err = set_ct(v_pulse, v_meas, number, p_width, interval);  
err = sweep_ct(range, capa, cond, time);  
. . .
```

See Also

Set_ct
Set_cv
Sweep_cv

Sweep_cv

This function performs C-G-V measurements on the CMU according to the conditions established by the Set_cv function, and returns error corrected values and bias values.

Synopsis

```
#include <pcs/tis.h>

int sweep_cv(range, capacitance, conductance, bias)
double range, capacitance[], conductance[], bias[];
```

Arguments

Item	Range Restrictions/Description
<i>range</i>	Specifies the measurement range. 0.0 for Auto range mode OR: Numeric expression [F] or [S] ¹ : 0 to 0.01
<i>capacitance</i>	A pointer to a double array where the measured capacitance values are stored. Any valid pointer [F]
<i>conductance</i>	A pointer to a double array where the measured conductance values are stored. Specify NULL if conductance is not needed (note that a NULL <i>bias</i> must also be specified). Any valid pointer [S]
<i>bias</i>	A pointer to a double array where the actual bias voltages are stored. Specify NULL if the bias value is not needed OR: Any valid pointer [V]

¹ If $0 < |\text{specified value}| \leq 1\text{E}-9$, the unit is farad F, and if $1\text{E}-9 < |\text{specified value}| \leq -2$, the unit is siemens S.

Description

This function returns capacitance and conductance measurement values and the bias values at which the measurements are made to the *capacitance*, *conductance*, and *bias* arrays, respectively.

The C-G measurement range is determined by the *range* value as follows:

Sweep_cv

Measurement Range Value	C Range (Maximum ¹)	G Range (Maximum ¹)
$0 < \text{value} \leq 1\text{E}-11$ $1\text{E}-9 < \text{value} \leq 1\text{E}-4$	10 pF (19.00 pF)	100 μS (120 μS)
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$ $1\text{E}-4 < \text{value} \leq 1\text{E}-3$	100 pF (190.0 pF)	1 mS (1.2 mS)
$1\text{E}-10 < \text{value} \leq 1\text{E}-9$ $1\text{E}-3 < \text{value} \leq 1\text{E}-2$	1000 pF (1900 pF)	10 mS (12 mS)
0	Auto range	

¹ The maximum value displayed on the front panel of the HP 4280A.

It is not necessary to define the exact range value for the HP 4280A; an arbitrary value is acceptable as long as it is within the allowable range.

The measurement resolution depends on the *integ_time* set in the Set_cmu function. Refer to the Set_cmu function.

Sweep_cv compensates for the effect of the cables that interconnect the CMU, SWM, and the pin boards. However, the measurement value includes the stray capacitance between the pin boards and the test device. First the stray capacitance must be measured without the DUT connected, then you must subtract the stray capacitance value from the measured capacitance value.

If an overflow occurs during a measurement, twice the value of the set range is returned. For example, if a measurement is performed on the 1000 pF/10 mS range and overflow occurs, 2E-9 and 2E-2 is returned to *capacitance* and *conductance*, respectively

The size of the *capacitance*, *conductance*, and *bias* arrays should be larger than the *number* specified in the Set_cv function, and -9999999.99999 is returned to the remaining array elements. If smaller, the measurement data that exceed the array size are written to an invalid memory location. This may cause the loss of some other variable content, a segment violation error, or a bus error.

In Offline Mode:

This function returns simulated measurement values to the *capacitance*, *conductance*, and *bias* arrays. Type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant Data Simulation mode:

This function returns a sequence of linear values to the *capacitance*, *conductance*, and *bias* arrays. Linear values returned to the *bias* array are determined by the *start*, *stop*, and *number* arguments of the Set_cv function. Linear values returned to the *capacitance* and *conductance* arrays range from 0 to the value determined by the *range* as described in the following table.

Measurement Range Value	Capacitance	Conductance
0	10 pF	100 μ S
$0 < \text{value} \leq 1\text{E}-11$	range value	100 μ S
$1\text{E}-11 < \text{value} \leq 1\text{E}-10$	range value	1 mS
$1\text{E}-10 < \text{value}$	range value	10 mS

File Data Simulation mode:

This function reads <c_meas>, <g_meas>, and <source> from the simulation file as the capacitance measurement, conductance measurement, and bias voltage values, respectively. The data simulation file format is as follows:

```
Sweep_cv
[
<c_meas>  <g_meas>  <source>
<c_meas>  <g_meas>  <source>
. . . .
<c_meas>  <g_meas>  <source>
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file.

Example

```
. . . .
int err;
int number;
double v_start, v_stop, hold, delay;
double range, capa[21], cond[21], bias[21];
. . . .
err = set_cv(v_start, v_stop, number, hold, delay);
err = sweep_cv(range, capa, cond, bias);
. . . .
```

See Also

Set_ct
Set_cv
Sweep_ct

Sweep_cv84

This function performs C-G-V measurements on the CMU84 according to the conditions established by the Set_cv84 function, and returns error corrected values and bias values.

Synopsis

```
#include <pcs/tis.h>

int sweep_cv84(range, capacitance, conductance, bias)
double range, capacitance[], conductance[], bias[];
```

Arguments

Item	Range Restrictions/Description
<i>range</i>	Specifies the measurement range by capacitance value. 0.0 for Auto range mode OR: Numeric expression [F]: — ¹
<i>capacitance</i>	A pointer to a double array where the measured capacitance values are stored. Any valid pointer [F]
<i>conductance</i>	A pointer to a double array where the measured conductance values are stored. Specify NULL if conductance is not needed (note that a NULL <i>bias</i> must also be specified). Any valid pointer [S]
<i>bias</i>	A pointer to a double array where the actual bias voltages are stored. Specify NULL if bias value is not needed OR: Any valid pointer [V]

¹ See the description of the *range* in the Measure_cmu84 function for details.

Description

The C-G-V measurement conditions are set by the Set_cv84 statement. The measurement resolution depends on the *integ_time* set in the Set_cmu84 function. Refer to the Set_cmu84 function.

The Sweep_cmu84 function translates the capacitance *range* to an impedance range, which the HP 4284A accepts. It is not necessary to define the exact range value for the HP 4284A; an arbitrary value is acceptable as long as it is within the allowable range.

Auto range mode occurs when the measurement *range* is set to zero. In Limited Auto range mode, you specify a measurement range and the specified measurement range is adopted during the sweep. See the Measure_cmu84 function for details.

The compensated capacitance data is returned to *capacitance*. When the measurement status is abnormal, -9999999.99999 is returned as the capacitance value.

10-30 Capacitance Measurements

The size of this array must be equal to or larger than the *number* set in the Set_cv84 function. If larger, the remaining elements of the array are filled with -9999999.99999 . If smaller, the measurement data that exceed the array size are written to an invalid memory location. This may cause the loss of some other variable content, a segment violation error, or a bus error.

The compensated conductance data is returned to the *conductance*. When the measurement status is abnormal, -9999999.99999 is returned as the capacitance value.

The size of this array must be the same as the size of the *capacitance*.

The step voltage is returned to the *bias* as the bias voltage. The size of this array must be the same size as the *capacitance*.

C-G-V measurement status: If the Port_status function is performed after executing the Sweep_cv84 function, the returned value reflects only the results of last sweep point.

In Offline Mode:

In offline mode execution, this function returns dummy data, which are a sequence of linear values. Linear values returned to the *bias* array are determined by the Set_cv84 statement. Linear values returned to the *capacitance* and *conductance* arrays range from 0 to C and G range values determined by the measurement *range*. C range means the value of the measurement *range* and G range means the value of the measurement *range* $\times 10^7$.

Example

```
. . .
int err;
int number;
double v_start, v_stop, hold, delay;
double range, capa[21], cond[21], bias[21];
. . .
err = set_cv84(v_start, v_stop, number, hold, delay);
err = sweep_cv84(range, capa, cond, bias);
. . .
```

See Also

Measure_cmu84
Set_cv84

Offline Debugging and Run-time Error Handling

Introduction

This chapter describes Offline debugging and Run-time Error handling tasks of the C Library. The functions are arranged alphabetically in this chapter.

Offline Debugging

The C Library provides the following functions for Offline debugging:

- Data_creation
- Is_on_line
- Off_line_data

These functions maximize system instrument efficiency, detect errors in the early program development stages without the need to access system instruments, and provide a method for developing measurement programs without an actual test device.

Run-time Error Handling

- Error_info
- Tis_errno

These functions return error numbers and error message strings.

Data_creation

This function creates an offline data simulation file.

Synopsis

```
#include <pcs/tis.h>

int data_creation(datafile)
char *datafile;
```

Argument

Item	Range Restrictions/Description
<i>datafile</i>	A pointer to a NULL terminated string specifying an offline simulation data file path name. Absolute or relative path accepted. Specify NULL to turn off the offline file simulation mode.

Description

This function starts and stops the automatic creation of an offline data simulation file.

Online mode operation

This function automatically creates an offline data simulation file and stores the measurement data of the TIS program into *datafile*.

Offline mode operation

Constant data simulation mode: This function automatically creates an offline data simulation file and stores the simulated constant values of the TIS program into *datafile*.

File data simulation mode: If you specified a *datafile*, this function resets to the constant data simulation mode, automatically creates an offline data simulation file. The simulated constant values of the TIS program are stored in the file.

File creation and data storing stops if this function executes with a NULL *datafile* or if a TIS error occurs. Also, file creation stops if the Off_line_data function executes while in the offline mode.

Example

```
. . .
if (data_creation("my_file") == -1 ) err_rep();
. . .
```

See Also

Off_line_data

Error_info

This function returns a TIS error number and an error message string.

Synopsis

```
#include <pcs/tis.h>

int err_info(errn, errm)
int errn[2];
char errm[100];
```

Arguments

Item	Range Restrictions/Description
<i>errn</i>	<p>A pointer to an integer array where the TIS error code and DCS error code (if available) is stored.</p> <p>errn[0]: TIS error number errn[1]: DCS error number (if <code>errn[0] = 40</code>) or CMU84 error number (if <code>errn[0] = 85</code>).</p> <p>Any valid pointer (size = 2)</p>
<i>errm</i>	<p>A pointer to a character array where the TIS error message is stored. A NULL terminated string is returned.</p> <p>Specify NULL to discard the message or any valid character array (size = 100).</p>

Description

This function examines the completion status of the most recently executed TIS function and returns the error numbers and error message string to *errn* and *errm*, respectively. The first element of the *errn* array represents the TIS error number. The second element represents the DCS error number if the TIS error number is 40 or the CMU84 error number if the TIS error number is 85.

If the previous TIS function executed with *no* errors, both error numbers are set to 0 and the error message string is a null string.

If the specified *errm* is not large enough to store the entire error message, the characters that exceed the array size are stored in an invalid memory location. This may cause the loss of some other variable content, a segment violation error, or a bus error. Be sure to maintain sufficient data space for both arguments.

This function returns no value by itself, even if an error occurs.

Example

```
. . .
error_info(error_number, error_msg);
. . .
```

Is_on_line

This function returns online test session status.

Synopsis

```
#include <pcs/tis.h>

int is_on_line()
```

Description

When running a test session, this function returns the online mode status. A 1 is returned as the function value if the session is in online mode. A 0 is returned if the session is in the offline mode. If an error is found, -1 is returned.

Example

```
. . .
int online;
. . .
if ((online=is_on_line())==-1) error_rep();
else {
    if (online==1) online_process();
    else offline_process();
}
. . .
```


Off_line_data

This function sets the offline data simulation mode.

Synopsis

```
#include <pcs/tis.h>

int off_line_data(datafile)
char *datafile;
```

Argument

Item	Range Restrictions/Description
<i>datafile</i>	<p>A pointer to a NULL terminated string specifying an offline simulation data file path name.</p> <p>Absolute or relative path accepted.</p> <p>Specify NULL to turn off the offline file simulation mode.</p>

Description

Calling this function with a *datafile* changes the offline data simulation mode to the file data simulation mode. In this mode, TIS measurement programs return the measurement data contained in the specified *datafile* as measurement results. Only an existing *datafile* is valid: specifying a nonexistent *datafile* results in a TIS error.

The file data simulation mode terminates if you load a program, link TIS functions or if you specify a NULL pointer to the *datafile*.

If an error is detected while reading a data simulation file, an error is reported and the offline data simulation mode is reset to the constant data simulation mode.

In the online mode, this function is ignored.

Example

```
. . .
if (off_line_data("DATA1") == -1 ) err_rep();
. . .
```

See Also

Data_creation

Tis_errno

The external variable `tis_errno` is set to the TIS error number whenever an error occurs in a TIS procedure.

Synopsis

```
#include <pcs/tis.h>

extern int tis_errno
```

Description

The external variable `tis_errno` is set to the TIS error number whenever an error occurs during a TIS function. The value is not changed if a TIS function ends successfully.

See Also

`Error_info`

Port and Unit Addresses

Introduction

The C Library provides the following functions to return the addresses of specified ports, units, or terminals:

- Fnaux
- Fncmh
- Fncml
- Fncmu
- Fncmu84
- Fngcmu
- Fngnd
- Fngsmu
- Fnport
- Fnsmu
- Fnunitsmu
- Fnvm
- Fnvs

The functions return the addresses for specified SWMs, SMUs, VSs, VMs, and CMUs.

These functions are arranged alphabetically in this chapter.

Fnaux

This function returns the address of the specified AUX (auxiliary) port of the SWM.

Synopsis

```
#include <pcs/tis.h>

int fnaux(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying an AUX port number. 1, 2, 3, 4, 11, 12, 13, 21, 22, 23, 31, 32, 33

Description

Port addresses, AUX port numbers, and AUX port names are as follows:

Port Name	AUX Port Number	Port Address	Port Name	AUX Port Number	Port Address
AUX1	1	32705	AUX3 (CMH)	3	32708
AUX2	2	32706	AUX4 (CML)	4	32709
AUX11 ¹	21	32721	AUX31 (CMH) ¹	27	32727
AUX12 ¹	22	32722	AUX32 ¹	28	32728
AUX13 ¹	23	32723	AUX33 (SMU) ¹	29	32729
AUX21 ¹	24	32724	AUX41 (CML) ¹	30	32730
AUX22 ¹	25	32725	AUX42 ¹	31	32731
AUX23 ¹	26	32726	AUX43 (SMU) ¹	32	32732

¹ For port expander only.

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .  
int anode;  
. . .  
anode = fnaux(1);  
. . .
```

See Also

Fnport

Fncmh

This function returns the CMH port address of the SWM and is functionally identical to `Fnport(8)`.

Synopsis

```
#include <pcs/tis.h>

int fncmh()
```

Description

This function can only be used if the CMH is connected to the AUX3 (CMH) port. If the SWM is equipped with a port expander, it can only be used if the CMH is connected to the AUX31 (CMH) port of the expander. Note that this is not identical to either `fnport(27)` or `fnaux(31)`.

This function does not report a TIS runtime error code. If the function is incorrectly used, `-1` is returned.

Example

```
. . .
int gate;
. . .
gate = fncmh();
. . .
```

See Also

Fncml
Fncmu
Fnport

Fncml

This function returns the CML port address of the SWM and is functionally identical to `Fnport(9)`.

Synopsis

```
#include <pcs/tis.h>

int fncml()
```

Description

This function can only be used if the CML is connected to the AUX4 (CML) port. If the SWM is equipped with a port expander, it can only be used if the CML is connected to the AUX41 (CML) port of the expander. Note that it is not identical to either `fnport(30)` or `fnaux(41)`.

This function does not report a TIS runtime error code. If the function is incorrectly used, `-1` is returned.

Example

```
. . .
int substrate;
. . .
substrate = fncml();
. . .
```

See Also

`Fncmh`
`Fncmu`
`Fnport`

Fncmu

This function returns the CMH or CML terminal address of the CMU.

Synopsis

```
#include <pcs/tis.h>

int fncmu(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying a CMU unit number. 1, 2

Description

The unit address corresponds to the *numb* as follows:

CMU Terminal	CMU <i>numb</i>	Unit Address
CMH terminal	1	32671
CML terminal	2	32672

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .
int mos_metal;
. . .
mos_metal = fncmu(1);
. . .
```

See Also

Fncmh
Fncml
Fnport

Fncmu84

This function returns the CMH or CML terminal address of the CMU84.

Synopsis

```
#include <pcs/tis.h>

int fncmu84(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying a CMU84 unit number. 1, 2

Description

The unit address corresponds to the *numb* as follows:

CMU Terminal	CMU84 <i>numb</i>	Unit Address
CMH terminal	1	32676
CML terminal	2	32677

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .
int mos_sub;
. . .
mos_sub = fncmu84(2);
. . .
```

See Also

Fncmh
Fncml
Fnport

Fngcmu

This function returns the GCMU port (Guard terminal of CMS) address of the SWM and is functionally identical to Fnport(11).

Synopsis

```
#include <pcs/tis.h>

int fngcmu()
```

Description

This function can only be used if the CMH and CML are connected to the AUX3 (CMH) and AUX4 (CML) ports, respectively. If the SWM is equipped with a port expander, this function can only be used if the CMH is connected to the AUX31 (CMH) port and if the CML is connected to the AUX41 (CML) port of the expander.

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .
int guard;
. . .
guard = fngcmu();
. . .
```

See Also

Fnport

Fngnd

This function returns the GNDU port address of the SWM and is functionally identical to Fnport(7).

Synopsis

```
#include <pcs/tis.h>

int fngnd()
```

Description

This function does not report a TIS runtime error code. If the function is incorrectly used, `-1` is returned.

Example

```
. . . . .
int emitter;
. . . . .
emitter = fngnd();
. . . . .
```

See Also

Fnport

Fngsmu

This function returns the GSMU port (guard of SMU1) address of the SWM and is functionally identical to Fnport(10).

Synopsis

```
#include <pcs/tis.h>

int fngsmu()
```

Description

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .
int guard;
. . .
guard = fngsmu();
. . .
```

See Also

Fnport

Fnport

This function returns the address of the specified SWM port. Port addresses are used to designate ports in TIS functions.

Synopsis

```
#include <pcs/tis.h>
```

```
int fnport(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying an SWM port position. −1 to 11, 21 to 32 ¹ , −8, −9

¹ For port expander only.

Description

Port number 0 is an imaginary port to which nothing is connected. Fnport(0) returns 0. Port numbers, port addresses, and port names are as follows:

Fnport

Port Name	Description	Port Number	Port Address
SMU1	SMU1 (for low current measurement)	1	32701
SMU2	SMU2	2	32702
SMU3	SMU3	3	32703
SMU4	SMU4	4	32704
AUX1	VS or VM ¹	5	32705
AUX2	VS or VM ¹	6	32706
AUX3 (CMH)	CMS High Port, VS or VM ¹	8	32708
AUX4 (CML)	CMS Low Port, VS or VM ¹	9	32709
AUX11 ²	VS or VM	21	32721
AUX12 ²	VS or VM	22	32722
AUX13 ²	VS or VM	23	32723
AUX21 ²	VS or VM	24	32724
AUX22 ²	VS or VM	25	32725
AUX23 ²	VS or VM	26	32726
AUX31 (CMH) ²	CMS High Port, VS or VM	27	32727
AUX32 ²	VS or VM	28	32728
AUX33 (SMU) ²	SMU	29	32729
AUX41 (CML) ²	CMS Low Port, VS or VM	30	32730
AUX42 ²	VS or VM	31	32731
AUX43 (SMU) ²	SMU	32	32732
GNDU	DC Ground Unit	7	32707
GSMU	Guard of SMU1	10, -1	32710
GCMU	CMS Guard Terminal	11, -8, -9	32711

¹ The units or terminals physically connected to the AUX1 through AUX4 ports are displayed during the pcsstart program under AUXILIARY PORT CONFIGURATION.

² For port expander only.

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .  
int collector, base;  
. . .  
collector = fnport(2);  
base = fnport(1);  
. . .
```

Fnsmu

This function returns the address of the specified SMU port of the SWM.

Synopsis

```
#include <pcs/tis.h>

int fnsmu(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying an SWM SMU port position. −1, 1 to 6, 29, 32

Description

This function is similar to the Fnport function. Refer to the Fnport function for details.

This function does not report a TIS runtime error code. If the function is incorrectly used, −1 is returned.

Example

```
. . .
int collector, base;
. . .
collector = fnsmu(2);
base = fnsmu(1);
. . .
```

Fnunitsmu

This function returns the unit address of the specified SMU. Unit addresses are used to designate SMUs in TIS programs.

Synopsis

```
#include <pcs/tis.h>

int fnunitsmu(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i> ¹	An integer value specifying SMU unit number. 1 to 8

¹ The SMU unit *numb* specifies the SMU. For example, *numb* = 2 specifies the second SMU from the left.

Description

The *numb* argument specifies the SMU (HP 41420A or HP 41421B) by the installed SMU order from the left. If the specified unit is an HP 41421B, this function returns a unit address that is in the range from 32621 to 32628. If the specified unit is an HP 41420A, this function returns a unit address that is in the range from 32629 to 32632.

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .
smu = fnunitsmu(3);
. . .
```


Fnvm

This function returns the unit address of the specified VM. Unit addresses are used to designate VMs in TIS programs.

Synopsis

```
#include <pcs/tis.h>
```

```
int fnvm(numb)
int numb;
```

Argument

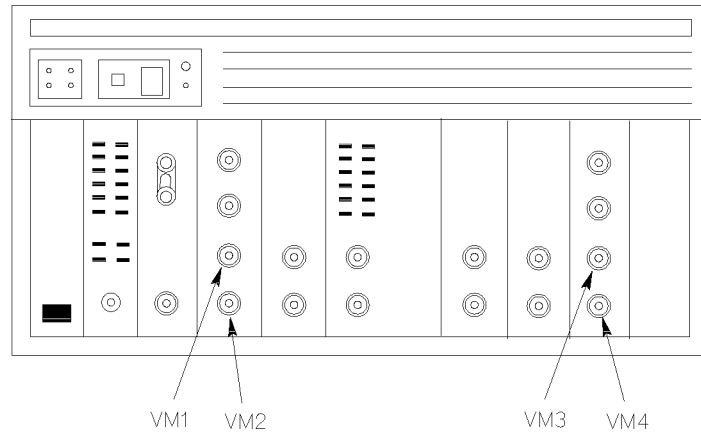
Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying a VM unit number. 1 to 16

Description

The *numb* argument specifies the VM (HP 41424A) unit number. This number corresponds to the installed VM position from the left as shown in the example on the next page. The unit addresses and corresponding VM unit numbers are as follows:

VM Unit Number	Unit Address	VM Unit Number	Unit Address
1	32634	9	32650
2	32636	10	32652
3	32638	11	32654
4	32640	12	32656
5	32642	13	32658
6	32644	14	32660
7	32646	15	32662
8	32648	16	32664

Fnmv



This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Example

```
. . .  
int term;  
. . .  
term = fnmv(2);  
. . .
```

See Also

Fnvs

Fnvs

This function returns the unit address of the specified VS. Unit addresses designate VSs in TIS programs.

Synopsis

```
#include <pcs/tis.h>

int fnvs(numb)
int numb;
```

Argument

Item	Range Restrictions/Description
<i>numb</i>	An integer value specifying a VS unit number. 1 to 16

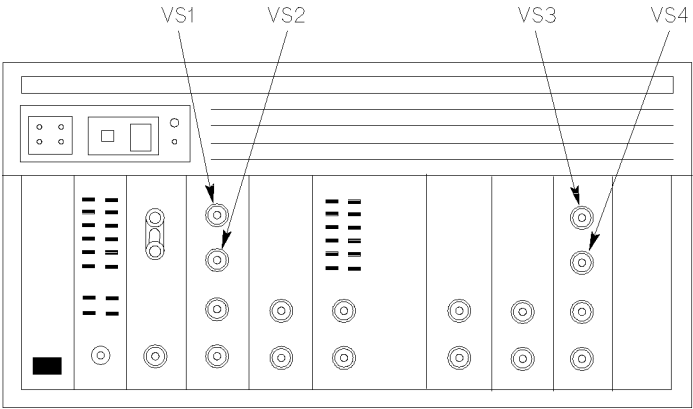
Description

The *numb* argument specifies the VS (HP 41424A) unit number. This number corresponds to the installed VS position from the left as shown in the example on the next page. The unit addresses and corresponding VS unit numbers are as follows:

VS Unit Number	Unit Address	VS Unit Number	Unit Address
1	32633	9	32649
2	32635	10	32651
3	32637	11	32653
4	32639	12	32655
5	32641	13	32657
6	32643	14	32659
7	32645	15	32661
8	32647	16	32663

This function does not report a TIS runtime error code. If the function is incorrectly used, -1 is returned.

Fnvs



Example

```
. . .  
int r_1;  
. . .  
r_1 = fnvs(1);  
. . .
```

See Also

Fnmv

C Library Prober Drivers

Introduction

This chapter provides C Library wafer prober control functions for the Electroglas 1034X, the Electroglas 2001X/2010X/3001X/4060X/4080X and the Tokyo Seimitsu A-PM-3000A/6000A/7000A/60A/80A/90A wafer probers.

Throughout this chapter, the following abbreviations are used to identify wafer probers:

EGX: Electroglas 1034X

EGY: Electroglas 2001X/2010X/3001X/4060X/4080X

TSK: Tokyo Seimitsu A-PM-3000A/6000A/7000A/60A/80A/90A

Caution



C Library prober control functions are written in C Language for the HP 4062UX, and are unprotected. They can, therefore, be accessed through the EDIT functions of the computer (such as the vi editor), making accidental erasure or modification of functions possible. To guard against this possibility, be sure to make backup copies of the system software before you begin using them.

Note



Some prober functions described in this chapter depend on the settings of manually operated prober switches. Refer to the manual of your prober for details.

Note



Each function describes all three supported prober drivers, unless otherwise noted.

The C Library provides the following functions for prober control:

- P_down
- P_home
- P_imove
- P_ink
- P_move
- P_orig
- P_pos
- P_scale
- P_up
- Prober_enter
- Prober_get_ba
- Prober_get_eid
- Prober_get_name
- Prober_identify
- Prober_init
- Prober_open
- Prober_output
- Prober_read_sysconfig
- Prober_reset
- Prober_spoll
- Prober_status
- Prober_wait
- Prober_waittime_ctl

These functions make prober control programs easier to write and can be used to control any wafer prober by simply linking the C Library.

The prober functions are arranged alphabetically in this chapter.

P_down

Control EGX, EGY, TSK

This function lowers the chuck of the wafer prober.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_down
```

Description

This function can be executed only when the EGX prober is set to Auto, or when the EGY prober or the TSK prober is set to Remote. If you include P_down in a program that also contains PCL functions, raise the wafer chuck before any PCL functions execute, especially if the PCL functions are nested in a loop.

For TSK, executing P_down releases the processor of the prober from a halt condition if a halt condition exists.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . .  
int ret;  
. . . .  
ret = p_down();  
. . . .
```

See Also

P_up

P_home

Control EGX, EGY, TSK

For EGX:

This function lowers the wafer chuck, moves the X-Y positioner to the home position, and sets the EGX prober to Manual.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_home()
```

Description

This function can be executed only when the EGX prober is set to Auto. When P_home is executed, the wafer chuck lowers, the X-Y positioner (if not presently at the home position) moves to the home position, and the prober is set to Manual. The operator can then perform manual operations, such as loading or unloading wafers.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .  
int ret;  
. . .  
ret = p_home();  
. . .
```


For EGY:

This function moves the X-Y stage of the prober to the wafer loading station, unloads the wafer, loads the next wafer, aligns the wafer, and probes the first probing chip position. Actual operation depends on the LOAD, PROFILE, and ALIGN switch settings of the prober. This function requires modifications if your EGY prober is *not* equipped with a profiler.

Synopsis

```
#include <pcs/prober.h>

int p_home()
```

Description

If the AUTO LOAD option of your prober is set to DISABLE, P_home instructs the prober to move the X-Y stage to the wafer loading station, at which time you can manually perform operations.

If LOAD, PROFILE, and ALIGN switch settings of your prober are set to ENABLE, P_home instructs the prober to move the X-Y stage to the wafer loading station, unload the present wafer, load the next wafer in the wafer stack (if the stack is not empty), profile and align the wafer, and move to the first probe position.

While performing P_home operations, the prober is set to Local (released from Remote) until alignment is complete. The prober must be in Remote, however, before any prober control functions can be executed.

Use the Prober_status function to determine whether the prober has returned to Remote. If the wafer stack is empty when P_home is executed, the current wafer is unloaded and a TIS ERROR 41 occurs.

If your EGY prober is *not* equipped with the AUTO LOAD, PROFILE, and/or AUTO ALIGN options or if any of these options are disabled, the EGY prober driver must be modified not to send the commands for the AUTOLOAD, PROFILE, or AUTO ALIGN options. For details, see the *Programming Guide for the C Library*.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is returned to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
. . .
ret = p_home();
. . .
```

P_home

For TSK:

This function moves the X-Y stage of the prober to the wafer loading station, unloads a wafer, loads the next wafer, and aligns the chuck at the initial probing position. Actual operation depends on the UNLOAD switch setting of the prober.

Synopsis

```
#include <pcs/prober.h>

int p_home()
```

Description

If the UNLOAD switch of the prober is set to MANUAL, P_home instructs the prober to move the X-Y stage to the wafer loading station. At this time you can perform any required manual operation. Also, the processor of the prober is released from its halt state and the prober is set to Local mode.

If the UNLOAD switch of your prober is set to AUTO, P_home instructs the prober to move the X-Y stage to the wafer loading station, unload the present wafer, load the next wafer in the wafer stack (if the stack is not empty), and align the X-Y stage at the initial probe position.

While performing P_home operations, the prober is set to Local (released from Remote) until alignment is complete. The prober must be in Remote, however, before any prober controlled functions can be executed.

Use the Prober_status function to determine whether the prober has returned to Remote. If the wafer stack is empty when P_home is executed, the present wafer unloads, the *last_wafer* flag (see Prober_status) is set to true, and your prober waits for further instructions.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
. . .
ret = p_home();
. . .
```

P_move

Control EGX, EGY, TSK

This function instructs the wafer prober to execute the relative movement of the X-Y stage. The X-Y stage moves the number of specified X- and Y-displacement units (established by P_scale) relative to the present X- and Y-coordinates.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_move(x, y)
double x, y;
```

Arguments

Item	Range Restrictions/Description
<i>x</i>	A double value that specifies the X displacement. Units: X-index set by P_scale $ X \text{ displacement} \times X\text{-index} \leq 999999.0$
<i>y</i>	A double value that specifies the Y displacement. Units: Y-index set by P_scale $ Y \text{ displacement} \times Y\text{-index} \leq 999999.0^1$

¹ For EGX, 99999.0 is available.

Description

This function can be executed only when the EGX prober is set to Auto, or when the EGY prober or the TSK prober is set to Remote. When P_move is executed, the X-Y stage of the prober moves in a direction opposite to that of the specified X- and Y-displacements.

For example, if both displacement values are positive, the X-Y stage moves toward the lower-left so that a position in the upper-right is aligned for probing. The chuck automatically lowers before the X-Y stage moves. For TSK, the processor of the prober is halted after the X-Y stage is moved.

If you use P_move in a program in conjunction with any PCL functions, the X-Y stage must be returned to its previous position before a PCL function can be executed.

This subroutine returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

P_imove

Example

```
. . .  
int ret;  
double x_dis, y_dis;  
. . .  
x_dis = 4.0;  
y_dis = 3.0;  
ret = p_imove(x_dis, y_dis);  
. . .
```

P_ink

Control EGY, TSK

For EGY:

This function triggers the specified inker of the wafer prober at its current position.

Synopsis

```
#include <pcs/prober.h>

int p_ink(inker_code)
int inker_code;
```

Argument

Item	Range Restrictions/Description
inker_code	An integer value that specifies the inker to be triggered. From 0 to 15

Description

P_ink can be executed only when your EGY prober is set to Remote.

This function inks the mark at the current position. The inkers are selected by the *inker_code* as shown below:

P_ink

<i>Inker_code</i>	Inker
0	none
1	1
2	2
3	1 2
4	3
5	1 3
6	2 3
7	1 2 3
8	4
9	1 4
10	2 4
11	1 2 4
12	1 2 4
13	1 3 4
14	2 3 4
15	1 2 3 4

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . .  
int ret;  
. . . .  
ret = p_ink(4);
```

For TSK:

This function triggers a specified inker of the wafer prober.

Synopsis

```
#include <pcs/prober.h>

int p_ink(inker_id)
int inker_id;
```

Argument

Item	Range Restrictions/Description
<i>inker_id</i>	An integer value that specifies the inker to be triggered. From 1 to 4

Description

P_ink can be executed only when your TSK prober is set to Remote. The P_ink function also releases the processor of the prober from its halt state.

This subroutine returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret, ink;
. . .
ink = 1;
ret = p_ink(ink);
. . .
```

P_move

Control EGX, EGY, TSK

This function instructs the wafer prober to execute an absolute movement of the X-Y stage. The X-Y stage moves to the specified X- and Y-coordinates.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_move(x,y)
double x, y;
```

Arguments

Item	Range Restrictions/Description
<i>x</i>	A double value that specifies the X coordinate. Units: X-index set by P_scale $ X \text{ displacement} \times X\text{-index} \leq 999999.0$
<i>y</i>	A double value that specifies the Y coordinate. Units: Y-index set by P_scale $ Y \text{ displacement} \times Y\text{-index} \leq 999999.0^1$

¹ For EGX, 99999.0 is available.

Description

This function can be executed only when the EGX prober is set to Auto or when the EGY prober or the TSK prober is set to Remote. When P_move is executed, the X-Y stage moves toward the lower left, if the specified X- and Y-coordinate values are larger than those of the present position. The chuck lowers automatically before the X-Y stage moves. P_move cannot be used in conjunction with any PCL functions.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
double x_co, y_co;
. . .
x_co = 4.0;
y_co = 3.0;
ret = p_move(x_co, y_co);
. . .
```


P_orig

Control EGX, EGY, TSK

This function assigns the specified X- and Y-coordinate values to the present position of the X-Y stage.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_orig(x,y)
double x,y;
```

Arguments

Item	Range Restrictions/Description
<i>x</i>	A double value that specifies the X coordinate. Units: X-index set by P_scale $ X \text{ displacement} \times X\text{-index} \leq 999999.0$
<i>y</i>	A double value that specifies the Y coordinate. Units: Y-index set by P_scale $ Y \text{ displacement} \times Y\text{-index} 999999.0^1$

¹ For EGX, 99999.0 is available.

Description

This function must be executed before any P_move or P_imove functions. The TSK prober must be set to Remote.

The present position of the X-Y stage is defined as the specified X- and Y-coordinates, and the specified X- and Y-coordinate values are stored as TIS internal variables.

If P_orig (0,0) executes when the X-Y stage is at (10,5), the TIS internal variables is set to (0,0) at (10,5), and the value displayed on the prober is 0,0.

P_orig cannot be used with Wstart_number or Wstart_xy PCL functions.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

P_orig

Example

```
. . .  
int ret;  
double x_orig, y_orig;  
. . .  
x_orig = 4.0;  
y_orig = 3.0;  
ret = p_orig(x_orig, y_orig);  
. . .
```

P_pos

Control EGX, EGY, TSK

This function returns the X- and Y-coordinates of the chip currently being probed.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_pos(x,y)
double *x, *y;
```

Arguments

Item	Range Restrictions/Description
<i>x</i>	A pointer to a double variable that specifies the X coordinate. Any valid pointer.
<i>y</i>	A pointer to a double variable that specifies the Y coordinate. Any valid pointer

Description

The X- and Y-coordinates returned by P_pos indicate the position of the chip currently being probed. These coordinates are stored in the corresponding prober driver variable and are updated whenever the X-Y stage is moved by a P_imove or P_move function.

The returned coordinates may differ from those of the actual position if either of the following conditions exist:

- If the EGX prober or the TSK prober is set to Manual, and the joystick is used to move the X-Y stage during the program.
- If the X- or Y-coordinate specified in the P_imove or P_move function causes the X-Y stage to move beyond the allowable range.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
double x_pos, y_pos;
. . .
ret = p_pos(&x_pos, &y_pos);
. . .
```

P_scale

Control EGX, EGY, TSK

This function sets the system index for the X- and Y-coordinates specified in the P_move and P_orig functions, and the system index for the X- and Y-displacements specified in the P_imove function.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_scale(x,y)
```

```
double x, y;
```

Arguments

Item	Range Restrictions/Description
<i>x</i>	A double value [μm] that specifies the X index. Any valid value
<i>y</i>	A double value [μm] that specifies the Y index. Any valid value

Description

The system index must be set by P_scale before a P_imove, P_move, or P_orig function can be executed. Normally, the system index needs to be set only after the wafer prober is first turned on.

All programmed movements of the X-Y stage are in the index units set by P_scale; joystick initiated movements, however, are in the index units set from either the subpanel or the operator console of the prober.

P_scale cannot be used in conjunction with W_start_number or W_start_xy PCL functions.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . . .
int ret;
double x_index, y_index;
. . . . .
x_index = 4000.0;
y_index = 3000.0;
ret = p_scale(x_index, y_index);
```

P_up

Control EGX, EGY, TSK

This function raises the chuck of the wafer prober.

Synopsis

```
#include <pcs/prober.h>
```

```
int p_up()
```

Description

This function can be executed only when the EGX prober is set to Auto, or the EGY prober or the TSK prober is set to Remote.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . .  
int ret;  
. . . .  
ret=p_up();  
. . . .
```

See Also

P_down

Prober_enter

Control EGX, EGY, TSK

This function enters data from a specified device and returns the length of the data.

Synopsis

```
#include "/usr/pcs/src/prober/prober_utl.h"
#include <sicl.h>

int prober_enter(eid, ba, buff, size)
INST eid;
int ba, size;
char *buff;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies the session ID obtained by Prober_open or Prober_get_eid.
<i>ba</i>	Specifies the bus address of a device that receives the data.
<i>buff</i>	Specifies a pointer to a buffer where the data is stored. The string includes a read termination character. A NULL character (0) is added at the end of the character string.
<i>size</i>	Specifies the size of the data buffer. If the buffer overflows, the length of the data string returned is -1 because of the NULL character.

Description

This function enters data from a specified device. The number of bytes of received data is returned if the data is received successfully. It does not count the trailing NULL character.

Data receiving is terminated in one of the following cases:

- A read termination character is received.
- A character with the EOI line asserted is received.
- A buffer overflow occurs.

In prober initializing functions (such as Prober_init), LF (‘\n’) is defined as a read termination character by default, using the SICL subroutine itermchr(). If you want to define another character as a read termination character, change the parameter of itermchr(). For details of this function, please refer to the *HP Standard Instrument Control Library Manual*.

For dull probers, this function can take a wait time to transfer data. A wait time can be set up to call `Prober_waittime_ctl`. If the wait time is set to 0, this function does not take a wait time. In this case, the fastest data transfer is executed. If you do not call `Prober_waittime_ctl`, a default value (100 μ seconds) is set up.

The number of bytes of received data is returned if the data is received successfully. If an error occurs, `-1` is returned and error number is saved to external variable `p_errno` to indicate the cause of the error. For a system call error, the error number is also returned to external variable `errno`. For a SICL function error, the error number is also returned by the `igeterrno()` function.

Example

```
. . .
INST eid;
int ret, ba;
char buff[21];
. . .
eid = prober_get_eid();
ba = prober_get_ba();
ret = prober_enter(eid, ba, buff, 20);
. . .
```

See Also

`Prober_init`
`Prober_output`

Prober_get_ba

Control —

This function returns the HP-IB bus address of the wafer prober opened by the Prober_init function.

Synopsis

```
#include <pcs/prober.h>

int prober_get_ba()
```

Description

This function returns the bus address of the wafer prober opened by the Prober_init function.

Example

```
. . .
int ba;
. . .
ba = prober_get_ba();
. . .
```

See Also

Prober_get_eid
Prober_get_name

Prober_get_eid

Control —

This function returns the session ID of the HP-IB interface opened by the Prober_init function.

Synopsis

```
#include <pcs/prober.h>
#include <sicl.h>

INST prober_get_eid(void)
```

Description

This function returns the session ID of the HP-IB interface opened by the Prober_init function.

Example

```
. . .
INST eid;
int ret;
. . .
ret = prober_init("EG1034X", 0, 0);
. . .
eid = prober_get_eid();
ret = iclose(eid);
. . .
```

Prober_get_name

Control —

This function returns the prober model name string.

Synopsis

```
#include <pcs/prober.h>

char *prober_get_name()
```

Description

This function returns the prober model name string that is specified by the Prober_init function, or that is chosen by the Prober_init function by examining the SYSCONFIG file. This function is helpful when writing a program that operates differently than one specified by the prober model.

This function returns a pointer to a NULL terminated string consisting of the prober model name.

Example

```
. . .
char *prober_name;
. . .
prober_name = prober_get_name();
if (strcmp(prober_name, "EG3001X") != 0 ) {
. . .
} else if (strcmp(prober_name, "TEL20S") != 0 ) {
. . .
}
. . .
```

Prober_identify

Control EGX, EGY, TSK

This function detects the existance of a specified prober.

Synopsis

```
#include <pcs/prober.h>

int prober_identify(scba)
int scba;
```

Argument

Item	Range Restrictions/Description
<i>scba</i>	Specifies the HP-IB logical unit number and bus address.

Description

This function opens the HP-IB interface by calling Prober_open, sets timeout to 0.1 second, and performs a serial poll on the specified HP-IB device. If the status byte message is returned, 0 is returned to indicate the existance of a prober. If the status byte message is not returned and a timeout occurs, -1 is returned.

For an HP-IB interface, the logical unit number is *scba* ÷ 100 and the bus address is *scba* % 100.

This function returns integer 0 if the specified prober is detected. If the prober does not exist or an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int exist, scba;
. . .
scba = 2703;
exist = prober_identity(scba);
. . .
```

Prober_init

Control EGX, EGY, TSK

This function initializes the prober and the prober drivers.

Synopsis

```
#include <pcs/prober.h>

int prober_init(prober_type, sc, ba)
char *prober_type;
int sc, ba;
```

Arguments

Item	Range Restrictions/Description
<i>prober_type</i>	A character string that specifies the type of prober. Range allowed: 30 characters max, including the termination character <code>\0</code> .
<i>sc</i>	Specifies a logical unit number. Range allowed: 0 and from 7 to 31.
<i>ba</i>	Specifies a bus address. Range allowed: 0 to 30.

Description

This function initializes a prober and its prober drivers specified by the *prober_type*. You must call this function before calling any prober driver functions.

The table below shows the prober name, the *prober_type*, and the Prober Library file.

Prober Name	<i>Prober_type</i>	Library File
ElectroGlas 1034X	“EGI034X” “EGX”	/usr/pcs/lib/libegx.a
ElectroGlas 2001X ElectroGlas 2010X ElectroGlas 3001X ElectroGlas 4060X ElectroGlas 4080X	“EG2001X” “EG2010X” “EG3001X” “EG4060X” “EG4080X” “EGY”	/usr/pcs/lib/libegy.a
Tokyo Seimitu APM3000A APM6000A APM7000A APM60A APM80A APM90A	“TSK” “APM3000A” “APM6000A” “APM7000A” “APM60A” “APM80A” “APM90A”	/usr/pcs/lib/libtsk.a

If both *sc* and *ba* are 0, this function searches the **SYSCONFIG** file for the address of a prober specified by *prober_type*, opens the HP-IB interface according to the address found in the **SYSCONFIG** file, detects the existence of a specified prober by performing a serial poll, and initializes the first prober found and its prober drivers.

When *prober_type* is a NULL string, this function searches through all the probers in the **SYSCONFIG** file and initializes the first prober found and its prober drivers.

If *sc* is not 0, this function opens the HP-IB interface in accordance with *sc* and initializes the prober specified by *ba* and its prober drivers. In this case, the **SYSCONFIG** file is not checked for the existence of probers.

If *prober_type* is DUMMY, this function initializes dummy prober drivers. In this case, *sc* and *ba* are “don’t care” values and are ignored.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
. . .
ret = prober_init("EG1034X", 27, 3);
. . .
```

See Also

Prober_output

Prober_open

Control —

This function opens the HP-IB interface according to a specified logical unit number and returns its session ID.

Synopsis

```
#include <pcs/prober.h>

INST prober_open(sc)
int sc;
```

Argument

Item	Range Restrictions/Description
<i>sc</i>	Specifies the logical unit number.

Description

This function opens the HP-IB interface according to the logical unit number *sc*.

The session ID of the open HP-IB interface is returned if successful. If the interface can not be opened, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
INST eid;
int sc;
. . .
sc = 8;
eid = prober_open(sc);
. . .
```

Prober_output

Control EGX, EGY, TSK

This function outputs data to a wafer prober initialized by the Prober_init function.

Synopsis

```
#include "/usr/pcs/src/prober/prober_utl.h"
#include <sic1.h>

int prober_output(eid, ba, buff, flag)
INST eid;
int ba, flag;
char *buff;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies the session ID of the open interface obtained by Prober_open or Prober_get_eid.
<i>ba</i>	An integer that specifies the bus address of the wafer prober where you send the data.
<i>buff</i>	A pointer to a buffer where the data are stored, or a character string to send. The string must be terminated by a NULL character ('\0').
<i>flag</i>	If <i>flag</i> is 0, CR ('\r') and LF ('\n') are sent as write termination characters after sending the string. If <i>flag</i> is 1, the last character of the string is sent with an EOI line asserted. The NULL character for string terminator is not sent to the wafer prober.

Description

This function outputs data to a wafer prober initialized by the Prober_init function.

For dull probers, this function can take a wait time to transfer data. A wait time can be set up to call Prober_waittime_ctl. If the wait time is set up to 0, this function does not take a wait time. In this case, the fastest data transfer occurs. If you do not call Prober_waittime_ctl, a default value (100 μ seconds) is used.

Prober_output returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Prober_output

Example

```
. . .  
INST eid;  
int ret, ba;  
. . .  
eid = prober_get_eid();  
ba = prober_get_ba();  
ret = prober_output(eid, ba, "?S", 0);  
. . .
```

See Also

Prober_enter
Prober_init

Prober_read_sysconfig

Control —

This function searches the `SYSCONFIG` file for a prober type and its address.

Synopsis

```
#include <pcs/prober.h>

int prober_read_sysconfig(prober_type, scba);
char *prober_type;
int *scba;
```

Arguments

Item	Range Restrictions/Description
<i>prober_type</i>	Specifies a pointer to a buffer where the character string of the prober type is stored.
<i>scba</i>	Specifies a pointer to a buffer where an integer, which indicates a logical unit number and bus address (for HP-IB interface), or logical unit number only (for RS-232C interface), is stored. For HP-IB: the logical unit number is $scba \div 100$ and the bus address is $scba \% 100$.

Description

This function searches the `SYSCONFIG` file for a prober type and its address.

An integer 1 is returned, if the data still remains in the `SYSCONFIG` file. If the `SYSCONFIG` file can not be read, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int eof, scba;
char pt[30];
. . .
while ((eof = prober_read_sysconfig(pt, &scba)) == 1) {
    if (strcmp(pt, "EG1034X") == 0) break;
}
if (eof==-1) err_rep();
. . .
```

Prober_reset

Control EGY, TSK

This function sets the prober to Local mode (releases the prober from Remote).

Synopsis

```
#include <pcs/prober.h>

int prober_reset()
```

Description

For the EGY prober, press the **PAUSE** key on the console of the EGY prober to recover from a prober error.

For the TSK prober, execute Prober_reset as the first step in recovering from a prober-related error.

When your prober is set to Local, manual operations from the control panel of the prober are possible.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
. . .
ret = prober_reset();
. . .
```

Prober_poll

Control EGX, EGY, TSK

This function performs a serial poll on a specified wafer prober and returns the status byte.

Synopsis

```
#include "/usr/pcs/src/prober/prober_utl.h"

int prober_poll(eid, ba)
INST eid;
int ba;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	An integer that specifies the session ID of the open HP-IB interface, obtained by Prober_open or prober_get_eid.
<i>ba</i>	An integer that specifies the bus address of the device being polled.

Description

This function performs a serial poll on a specified wafer prober.

An integer, the lowest byte that contains the status byte message from the specified device, is returned. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
INST eid;
int status, ba;
. . .
eid = prober_get_eid();
ba = prober_get_ba();
status = prober_poll(eid, ba);
. . .
```

Prober_status

Control EGX, EGY, TSK

This function returns three integer values to indicate the status of the wafer prober.

Synopsis

```
#include <pcs/prober.h>

int prober_status(remote, on_wafer, last_wafer)
int *remote, *on_wafer, *last_wafer;
```

Arguments

Item	Range Restrictions/Description
<i>remote</i>	A pointer to an integer that specifies a variable name. Any valid pointer.
<i>on_wafer</i>	A pointer to an integer that specifies a variable name. Any valid pointer.
<i>last_wafer</i> ¹	An integer that specifies a variable name. Any valid name.

¹ *Last_wafer* is a dummy parameter for EGX and EGY.

Description

The values returned are either 1 or 0. When a condition is true, 1 is returned; otherwise 0 is returned.

Variable Name	Returned Value	Description
<i>remote</i>	1	The wafer prober is controlled by HP 4062UX using the Prober Driver Library (prober in Auto/Remote).
	0	The wafer prober is controlled from subpanel (prober in Manual/Local).
<i>on_wafer</i>	1	On the wafer
	0	Not on the wafer (the wafer edge is detected).
<i>last_wafer</i>	1	The last wafer is unloaded.
	0	There are wafers to be loaded.

The *on_wafer* value may change as a result of a P_move, P_imove, or P_up initiated X-Y stage movement. The *last_wafer* value is set to 0 as long as wafers remain in the wafer stack, and is set to 1 after the last wafer is unloaded from the chuck following a P_home function. The *last_wafer* value is always 0 if your TSK prober is *not* equipped with an automatic wafer loader.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system

call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . .  
int ret;  
int remote, on_wafer, last_wafer;  
. . . .  
ret = prober_status(&remote, &on_wafer, &last_wafer);  
. . . .
```

Prober_wait

Control —

This function waits for a specified period.

Synopsis

```
#include "/usr/pcs/src/prober/prober_utl.h"

int prober_wait(time)
double time;
```

Argument

Item	Range Restrictions/Description
<i>time</i>	Specifies a period to wait. The units are in microseconds. Resolution: 100 microseconds

Description

This function waits for a period of time.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *p_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
double wait_time;
. . .
wait_time = 3000.0;
ret = prober_wait(wait_time);
. . .
```

Prober_waittime_ctl

Control —

This function sets up a wait time for transferring data through the HP-IB interface.

Synopsis

```
#include "/usr/pcs/src/prober/prober_utl.h"

void prober_waittime_ctl(time)
double time;
```

Argument

Item	Range Restrictions/Description
<i>time</i>	Specifies a wait time. The units are in microseconds. Resolution: 100 microseconds

Description

This function sets up a specified time to wait before transferring data through the HP-IB interface. If you specify 0 for *time*, the data are transferred with no delay. If this function is not called, a default value (100 microseconds) is set.

Example

```
. . .
double wait_time;
. . .
wait_time = 3000.0;
prober_waittime_ctl(wait_time);
. . .
```


Probing Control and File Creation Libraries

Introduction

This chapter describes the Probing Control Library (PCL) and the File Creation Library (FCL) that are included in the C Library. The PCL and FCL functions are arranged alphabetically in this chapter.

The Probing Control Library (PCL)

The Probing Control Library (PCL) is a set of functions that are used to control an automatic wafer prober. The functions are used with the Probing Pattern Data file created by the Probing Pattern Generator (PPG). PCL in the C Library supports files on the HFS format only.

- W_get
- W_move_next
- W_move_number
- W_move_xy
- W_pos_number
- W_pos_xy
- W_return_limit
- W_return_number
- W_return_xy
- W_start_number
- W_start_xy
- W_total_chip

The PCL functions get probing pattern data files, designate initial probing positions, specify chip and test modules to probe, and report the present probing position. There are ten PCL functions, all of which make it easy to control the prober.

The File Creation Library (FCL)

The File Creation Library (FCL) is a set of functions that create data files for wafer maps and statistical graphics. The measurement data files created by the C Library FCL are compatible with the basic statistics and data manipulation (BSDM) system. You can also create a data file formatted for DOS text file, which can be operated by any database and data analysis software.

- W_build_file
- W_file_type
- W_put_array
- W_put_data
- W_read_file
- W_save_data

There are six functions in the FCL and there are three ways to create a measurement data file.

W_build_file

This function generates and saves multiple parameter measurement data files.

Synopsis

```
#include <pcs/wafer.h>

int w_build_file(file_specifier, data_array, number_data,
                file_comments, variable_name1,
                variable_name2, ..., variable_name10)
char *file_specifier, *file_comments, *variable_name1,
    *variable_name2, ..., *variable_name10;
double *data_array;
int number_data;
```

Arguments

Item	Range Restrictions/Description
<i>file_specifier</i>	A character string that specifies the name of a measurement data file and may include a path name. Maximum length: 1023 characters.
<i>data_array</i>	A two-dimensional array that specifies a pointer to an array of measurement data. Any valid pointer.
<i>number_data</i>	Specifies the number of data of the measurement data array pointed to by <i>data_array</i> . Value corresponds to the number of elements of the second subscript.
<i>file_comments</i>	A character string that specifies a title or comments for a measurement data file. Maximum length: 80 characters
<i>variable_name1</i> , ..., <i>variable_name_10</i>	Specify names of variables. These names correspond to each element of the first subscript of the measurement data array pointed to by <i>data_array</i> . Up to 10 variables with 10 characters max for each variable name. ¹

¹ The *variable_names* correspond to rows in the *data_array*. Use a NULL pointer if all ten *variable_names* are not needed.

W_build_file

Description

This function converts the *data_array* that you set up in a BSDM format data file and stores the data file, *file_comments*, and *variable_names* in a specified file. Each row of *data_array* is assigned to a *variable_name* by the *W_build_file*, so make sure that you set up the *data_array* correctly.

This function can be used by itself to create a data file. The *data_array* created by this function is in the same format as the BSDM system and can be used to generate wafer maps and statistical graphics. You can also create a DOS text file and store the measured data in it. The *W_file_type* decides the data file type: BSDM binary or DOS text.

Numeric arrays used with this function must be specified as follows:

```
data_array_name[number_of_variable_names]
                [maximum_number_of_data_per_variable]
```

	1	2	3	4 (maximum number of data)
Variable name 1	Data	-----			
Variable name 2		-----			

(max. 10		-----			
variables)		-----			

example: `data_array[3][150];`

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . . .
int ret;
double value[3][50];
. . . .
ret = w_build_file("/test/prober/file1", value,
                  50, "MEASUREMENT DATA", "Vth",
                  "hFE", "Cox", NULL);
. . . .
```

W_file_type

This function specifies the data file type: BSDM binary or DOS text file. The W_save_data and W_build_file functions create data files according to this file type setting.

Synopsis

```
#include <pcs/wafer.h>

int w_file_type(file_type)
char *file_type;
```

Argument

Item	Range Restrictions/Description
<i>file_type</i>	A character string that specifies the file type. "BDAT" and "DOS" are available.

Description

The W_file_type function specifies the data file type. When W_file_type is executed with the parameter "DOS" specified, the data file type is set to DOS text file. When W_file_type is executed with the parameter "BDAT" specified, the data file type is set to BDAT file formatted for the BSDM software.

The W_save_data and W_build_file functions create data files determined by the data file type.

The data file type is set to BDAT initially.

For more information on the DOS text file, see *System Operation*.

Example

```
. . .
int ret;
. . .
ret = w_file_type("DOS");
. . .
ret = w_save_data(. . . .);
. . .
```

W_get

This function retrieves probing data from a specified probing pattern data file.

Synopsis

```
#include <pcs/wafer.h>

int w_get(file_specifier)
char *file_specifier;
```

Argument

Item	Range Restrictions/Description
<i>file_specifier</i>	A character string that specifies the name of a probing pattern data file and may include a path name. Maximum length: 1023 characters

Description

The file specified in this function must be a probing pattern data file previously generated by PPG. This function loads the probing data from the specified file to the internal data structure. Therefore, this function must be executed in a measurement program before any other PCL functions can be executed.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
char *file;
. . .
ret = w_get("file1");
. . .
ret = w_get("/test/prober/file2");
. . .
file = "file3";
ret = w_get(file);
```

W_move_next

This function moves the wafer probe chuck to the next test module in accordance with the set probing sequence.

Synopsis

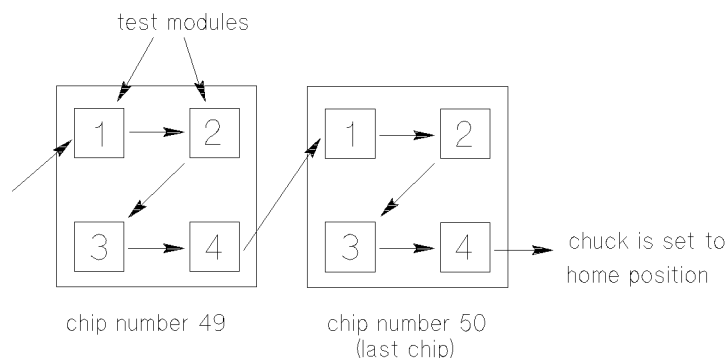
```
#include <pcs/wafer.h>
```

```
int w_move_next()
```

Description

When this function is executed, the wafer probe chuck is set to the next test module specified by the probing sequence established in the probing pattern data file. When the last test module on a wafer is probed, the wafer probe chuck moves to the home position (P_home function is executed by this function).

For example, if the probing sequence shown in the following figure is specified, the wafer probe probes the next test module each time W_move_next is executed.



The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_move_next can be executed.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
. . .
int ret;
. . .
ret = w_move_next();
. . .
```

W_move_number

This function moves the wafer probe chuck to the specified test module on the specified chip.

Synopsis

```
#include <pcs/wafer.h>

int w_move_number(chip_number, module_number)
int chip_number, module_number;
```

Arguments

Item	Range Restrictions/Description
<i>chip_number</i>	Specifies the number of a chip. It can be 1 to the total selected chip number. Integer expression, value set by PPG
<i>module_number</i>	Specifies the number of a module. Integer expression, value set by PPG

Description

The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_move_number can be executed.

The *module_number* can be the minimum module number to the maximum module number. The total selected chip number, the minimum module number and the maximum module number should be previously stored into an internal data structure from a probing pattern data file selected by W_get. If there is only one module in a chip, *module_number* must be 1.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int chip, module;
. . .
ret = w_move_number(1, 1);
. . .
chip = 2;
module = 5;
ret = w_move_number(chip, module);
```


W_move_xy

This function moves the wafer probe chuck to the specified test module on the chip at the specified X-Y coordinates.

Synopsis

```
#include <pcs/wafer.h>

int w_move_xy(x_chip_coordinate, y_chip_coordinate, module_number)
int x_chip_coordinate, y_chip_coordinate, module_number;
```

Arguments

Item	Range Restrictions/Description
<i>x_chip_coordinate</i>	Specifies the X-chip coordinate of a chip. Integer expression, value set by PPG
<i>y_chip_coordinate</i>	Specifies the Y-chip coordinate of a chip. It can be 1 to the maximum X-Y chip coordinates. Integer expression, value set by PPG
<i>module_number</i>	Specifies the number of a module. Integer expression, value set by PPG

Description

The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_move_xy can be executed.

The *module_number* can be the minimum module number to the maximum module number. The maximum X-Y chip coordinates, and the minimum and maximum module number should be stored into an internal data structure from a probing pattern data file selected by W_get. If there is only one module in a chip, the *module_number* must be 1.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int chip_x, chip_y, module;
. . .
ret = w_move_xy(2, 2, 1);
. . .
chip_x = 3; chip_y = 5; module = 10;
ret = w_move_xy(chip_x, chip_y, module);
```

W_pos_number

This function returns the chip number and test module number of the present wafer prober chuck position.

Synopsis

```
#include <pcs/wafer.h>

int w_pos_number(chip_number, module_number)
int *chip_munber, *module_number;
```

Arguments

Item	Range Restrictions/Description
<i>chip_number</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>module_number</i>	Specifies a pointer to an integer variable. Any valid pointer

Description

The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_pos_number can be executed.

When a *module_number* is not needed and a NULL pointer is set as the second parameter, a value is not returned.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int chip, module;
. . .
ret = w_pos_number(&chip, &module);
. . .
ret = w_pos_number(&chip, NULL);
```

W_pos_xy

This function returns the X- and Y-chip coordinates and the test module number of the present wafer prober chuck position.

Synopsis

```
#include <pcs/wafer.h>

int w_pos_xy(x_chip_coordinate, y_chip_coordinate, module_number)
int *x_chip_coordinate, *y_chip_coordinate, *module_number;
```

Arguments

Item	Range Restrictions/Description
<i>x_chip_coordinate</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>y_chip_coordinate</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>module_number</i>	Specifies a pointer to an integer variable. Any valid pointer

Description

The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_pos_xy can be executed.

When a *module_number* is not needed and a NULL pointer is set as the second parameter, a value is not returned.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int chip_x, chip_y, module;
. . .
ret = w_pos_xy(&chip_x, &chip_y, &module);
. . .
ret = w_pos_xy(&chip_x, &chip_y, NULL);
```

W_put_array

This function transfers a measurement *data_array* to the internal data structure and assigns a *variable_name* to the data array. Block data arrays of the internal data structure can be combined into a measurement data file with the W_save_data function.

Synopsis

```
#include <pcs/wafer.h>

int w_put_array(variable_name, data_array, number_data)
char *variable_name;
double *data_array;
int number_data;
```

Arguments

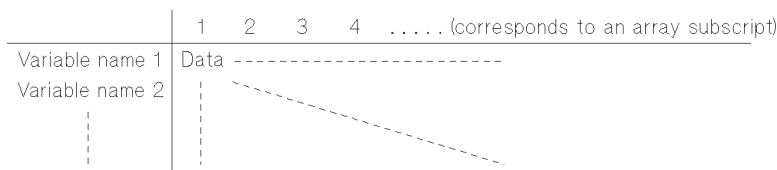
Item	Range Restrictions/Description
<i>variable_name</i>	Specifies the name of a variable where measurement data is stored. Maximum length: 10 characters
<i>data_array</i>	Specifies a pointer to an array of measurement data. Any valid pointer
<i>number_data</i>	Specifies the number of data of the measurement data array pointed to by <i>data_array</i> . This integer corresponds to an array subscript.

Description

The W_put_array function transfers a user-specified one-dimensional measurement *data_array* to the internal data structure of the system, where it is assigned the *variable_name*.

When the W_save_data function is executed, data arrays in the internal data structure are combined to create a measurement data file. Refer to the W_save_data function. A maximum of 500 variable names can be contained in a data file.

The data file created by the W_save_data function has the structure shown in the following figure:



Note

W_put_array and W_put_data cannot be used in the same measurement program.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
char *name;
double value[20];
int num_data;
. . .
ret = w_put_array("Vth", value, 20);
. . .
name = "Vth";
num_data = 20;
ret = w_put_array(name, value, num_data);
```

W_put_data

This function transfers a *measurement_value* to the internal data structure and assigns it to the specified *variable_name*. Measurement values of the internal data structure assigned to various variable names can be combined into a measurement data file by the W_save_data function.

Synopsis

```
#include <pcs/wafer.h>

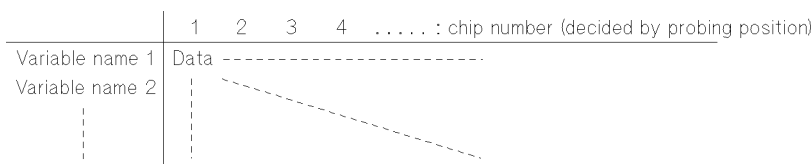
int w_put_data(variable_name, data)
char *variable_name;
double data;
```

Arguments

Item	Range Restrictions/Description
<i>variable_name</i>	A string that specifies a name of a variable in which measurement data are stored. Maximum length: 10 characters Maximum number: 500 in a measurement data file
<i>data</i>	A numeric expression that specifies the value of measurement data.

Description

The W_put_data function transfers the *measurement_value* to the internal data structure where it is assigned to the specified *variable_name* as shown in the following figure:



When this function is used, it must be executed after each measurement. After using W_put_data functions to transfer all the desired measurement values to the desired variable names in the internal data structure, execute the W_save_data function to create a measurement data file from this data. Refer to W_save_data. This method is ideal for creating measurement data files for wafer mapping, because the measurement values can easily be arranged by chip number order by using the chip number as the loop counter.

A maximum of 500 variable names can be contained in a data file. The maximum number of *data* is the total selected chip number previously stored into an internal data structure from the probing pattern data file selected by W_get.

The W_get function and the W_start_xy or W_start_number function must be executed in a measurement program before W_put_data can be executed.

Note

W_put_array and W_put_data cannot be used in the same measurement program.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
char *name;
double value;
. . .
ret = w_put_data("Vth", value);
. . .
name = "Vth";
ret = w_put_data(name, value);
```

W_read_file

This function reads the BSDM data file, which is created by the FCL functions. The data is retrieved in the `fcl_struct` data structure.

Synopsis

```
#include <pcs/wafer.h>

int w_read_file(filename, data_struct)
char *filename;
struct fcl_struct *data_struct;
```

Argument

Item	Range Restrictions/Description
<i>filename</i>	A character string that specifies the name of a BSDM format file. Maximum length: 1023 characters
<i>data_struct</i>	A pointer of <code>fcl_struct</code> data structure. The data structure must be previously defined.

Description

This function reads the BSDM data file, which is created by FCL functions, and stores the data in the `fcl_struct` data structure. The `fcl_struct` is declared in the `wafer.h` header file, as follows:

```
#define FCL_MAX_VAR_NUM 500

struct fcl_struct {
    char file_comments[81];
    char var_name[FCL_MAX_VAR_NUM][11];
    double *data[FCL_MAX_VAR_NUM];
    double var_num;
    double obs_num;
    double data_num;
};
```

This function returns integer 0 if successful. If an error occurs, `-1` is returned and error number is saved to external variable `w_errno` to indicate the cause of the error. For a system call error, the error number is also returned to external variable `errno`. For a SICL function error, the error number is also returned by the `igeterrno()` function.

Example

```
struct fcl_struct meas_data;
int ret;
char *file;
. . .
ret = w_read_file("/users/test/meas/data1", meas_data);
. . .
file = "data_x";
ret = w_read_file(file, meas_data);
. . .
```

See Also

W_build_file
W_save_data

W_return_limit

This function returns the maximum and minimum values of the X- and Y-chip coordinates from the probing pattern data created by PPG.

Synopsis

```
#include <pcs/wafer.h>

int w_return_limit(min_x_chip, max_x_chip, min_y_chip, max_y_chip)
int *min_x_chip, *max_x_chip, *min_y_chip, *max_y_chip;
```

Arguments

Item	Range Restrictions/Description
<i>min_x_chip</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>max_x_chip</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>min_y_chip</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>max_y_chip</i>	Specifies a pointer to an integer variable. Any valid pointer

Description

The W_get function must be executed in a measurement program before this function can be executed.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int x_min, x_max, y_min, y_max;
. . .
ret = w_return_limit(&x_min, &x_max, &y_min, &y_max);
```

W_return_number

This function returns a chip number corresponding to the specified X-Y coordinates. This is the inverse function of the W_return_xy function.

Synopsis

```
#include <pcs/wafer.h>

int w_return_number(x_chip_coordinate, y_chip_coordinate)
int x_chip_coordinate, y_chip_coordinate;
```

Arguments

Item	Range Restrictions/Description
<i>x_chip_coordinate</i>	Specifies the X-chip coordinate of a chip. Integer expression, value set by PPG
<i>y_chip_coordinate</i>	Specifies the Y-chip coordinate of a chip. It can be 1 to the maximum X-Y chip coordinates. Integer expression, value set by PPG

Description

This function returns the chip number corresponding to the X- and Y-coordinates from the probing pattern data created by PPG. The W_get function must be executed in a measurement program before this function can be executed.

This procedure always returns a positive number when it successfully completes. A return value of -1 must be interpreted as an error occurrence and the error code is stored in *w_errno*.

Example

```
int number;
int chip_x, chip_y;
. . .
number = w_return_xy(1, 2);
. . .
chip_x = 10;
chip_y = 20;
number = w_return_xy(chip_x, chip_y);
```

W_return_xy

This function returns the X- and Y-chip coordinates corresponding to the specified chip number.

Synopsis

```
#include <pcs/wafer.h>
```

```
int w_return_xy(x_chip_coordinate, y_chip_coordinate, chip_number)
int *x_chip_coordinate, *y_chip_coordinate, chip_number;
```

Arguments

Item	Range Restrictions/Description
<i>x_chip_coordinate</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>y_chip_coordinate</i>	Specifies a pointer to an integer variable. Any valid pointer
<i>chip_number</i>	Specifies the number of a chip. It can be 1 to the total selected chip number set by PPG. Integer expression

Description

This function returns the X- and Y-coordinates corresponding to the specified chip number. The probing pattern data should be previously stored into an internal data structure by W_get.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
int chip_number, chip_x, chip_y;
. . .
ret = w_return_xy(&chip_x, &chip_y, 20);
. . .
chip_number = 50;
ret = w_return_xy(&chip_x, &chip_y, chip_number);
```

W_save_data

This function generates measurement data files in conjunction with the W_put_data or W_put_array functions.

Synopsis

```
#include <pcs/wafer.h>

int w_save_data(file_specifier, file_comments)
char *file_specifier, *file_comments;
```

Arguments

Item	Range Restrictions/Description
<i>file_specifier</i>	A character string that specifies the name of a measurement data file and may include a path name. Maximum length: 1023 characters.
<i>file_comments</i>	A character string that specifies a title or comments for a measurement data file. Maximum length: 80 characters

Description

W_save_data creates a measurement data file as the specified file name, and then transfers the data that is stored in the internal data structure by the W_put_data or W_put_array function to this data file, along with any *file_comments*. The data files created by this function follow the same format as the BSDM system, and can be used to generate wafer maps and statistical graphs. You can also create a DOS text file and store the measured data in it. The W_file_type decides the data file type: BSDM binary or DOS text.

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;
char *file, *title;
. . .
ret = w_save_data("/test/prober/file1", "MEASUREMENT DATA-1");
. . .
file = "file2";
title = "This is a measurement data file.";
ret = w_save_data(file, title);
```

W_start_number

This function determines the wafer prober starting position.

Synopsis

```
#include <pcs/wafer.h>

int w_start_number(chip_number, module_number)
int chip_number, module_number;
```

Arguments

Item	Range Restrictions/Description
<i>chip_number</i>	Specifies the number of a chip. It can be 1 to the total selected chip number set by PPG. Integer expression
<i>module_number</i>	Specifies a number of a module set by PPG. Integer expression, 1 to the total selected module number set by PPG.

Description

The *chip_number* and *module_number* specified in this function must agree with the actual position of the wafer prober before probing begins. For example, if chip 1, module 2 is specified in `W_start_number`, the prober must be positioned at chip 1, module 2.

Module_number can be the minimum module number to the maximum module number. The probing pattern data should be stored previously into an internal data structure by `W_get`.

If there is only one module in a chip, *module_number* must be 1.

`W_start_number` or `W_start_xy` must be executed in a measurement program before any of the following functions can be executed:

- `W_pos_number`
- `W_pos_xy`
- `W_move_next`
- `W_move_number`
- `W_move_xy`

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;  
int chip, module;  
.  
.  
ret = w_start_number(1, 1);  
.  
.  
chip = 2;  
module = 5;  
ret = w_start_number(chip, module);
```

W_start_xy

This function determines the wafer prober starting position.

Synopsis

```
#include <pcs/wafer.h>
```

```
int w_start_xy(x_chip_coordinate, y_chip_coordinate, module_number)
int x_chip_coordinate, y_chip_coordinate, module_number;
```

Arguments

Item	Range Restrictions/Description
<i>x_chip_coordinate</i>	Specifies the X-coordinate of a chip. Integer expression
<i>y_chip_coordinate</i>	Specifies the Y-coordinate of a chip. Integer expression
<i>module_number</i>	Specifies the number of a module. Integer expression

Description

The *x_* and *y_chip_coordinates* and *module_number* specified in this function must agree with the actual position of the water prober before probing begins, as in the *W_start_number* function.

The probing pattern data should be previously stored into an internal data structure by *W_get*.

The *module_number* can be the minimum module number to the maximum module number. If there is only one module in a chip, *module_number* must be 1.

W_start_xy or *W_start_number* must be executed in a measurement program before any of the following functions can be executed:

- W_pos_number*
- W_pos_xy*
- W_move_next*
- W_move_number*
- W_move_xy*

This function returns integer 0 if successful. If an error occurs, -1 is returned and error number is saved to external variable *w_errno* to indicate the cause of the error. For a system call error, the error number is also returned to external variable *errno*. For a SICL function error, the error number is also returned by the *igeterrno()* function.

Example

```
int ret;  
int chip_x, chip_y, module;  
.  
.  
ret = w_start_xy(2, 2, 1);  
.  
.  
chip_x = 3;  
chip_y = 5;  
module = 10;  
ret = w_start_xy(chip_x, chip_y, module);
```

W_total_chip

This function returns the number of chips per wafer registered by PPG.

Synopsis

```
#include <pcs/wafer.h>

int w_total_chip()
```

Description

This function returns the total number of chips from the probing pattern data created by PPG. The W_get function must be executed in a measurement program before this function can be executed.

This procedure always returns a positive number when it successfully completes. A return value of -1 must be interpreted as an error occurrence and the error code is stored in *w_errno*.

Example

```
. . .
int number;
. . .
number = w_total_chip();
```

DCS Program Memory Library

Introduction

The C Library provides the following functions for the HP 4142B DCS Program Memory Library:

- Fnaddr4142
- Fnchannel4142
- Fnrange4142
- Fnunit4142
- Pcs_hpib_clear
- Pcs_hpib_enter
- Pcs_hpib_open
- Pcs_hpib_output
- Pcs_hpib_spoll
- Pgm_ch_sw_off
- Pgm_ch_sw_on
- Pgm_disable_dcs
- Pgm_end
- Pgm_force_i
- Pgm_force_v
- Pgm_measure_i
- Pgm_measure_p
- Pgm_measure_v
- Pgm_search_iv
- Pgm_set_asearch
- Pgm_set_pbias
- Pgm_set_smu
- Pgm_set_vm
- Pgm_start
- Pgm_wait
- Readererror4142
- Readout_dcs
- Readsweep_dcs
- Run_dcs_program

The DCS Program Memory Library improves the total system throughput of the HP 4062UX system. The HP 4142B serves as a data storage area for a sequence of frequently used control commands. Triggering the stored program requires much less time than re-entering the command sequences each time they are used.

The functions are arranged alphabetically in this chapter.

Fnaddr4142

This function returns the DCS device selector, which consists of an interface logical unit number and a primary address.

Synopsis

```
#include <pcs/tis.h>

int fnaddr4142()
```

Description

This function gets the device selector of the DCS. The device selector consists of an interface logical unit number (*sc*) and a primary bus address (*ba*). For the default system configuration, this function returns 2723.

If the DCS is not configured, turned off, or if the system is in the offline mode, this function returns 0.

Example

```
. . .
int dcs;
. . .
dcs=fnaddr4142();
```

Fnchannel4142

This function returns the DCS channel number of a specified unit address.

Synopsis

```
#include <pcs/tis.h>

int fnchannel4142(unit)
int unit;
```

Argument

Item	Range Restrictions/Description
<i>unit</i>	Specifies the address of the DCS unit. Integer expression: 32621 to 32664

Description

This function is useful for storing the DCS control commands into program memory. When you use a program that sends the DCS control commands using the Pcs_hpib_output function, the channel number of the DCS (1 through 8, 11 through 18, 21 through 28) is used to specify the units under control (SMU, VS, or VM). You do not need to know where the units are installed in the DCS to use this function.

If there is no unit available at the *unit* address, the function returns 0.

For more information on the channel number, see the HP 4142B *Operation Manual*.

Example

```
. . .
int ch_smu1;
. . .
ch_smu1=fnchannel4142(32621);
ch_smu1=fnchannel4142(fnunit4142(fnsmu(3)));
. . .
```

See Also

Fnunit4142

Fnrange4142

This function returns the DCS range code of a specified value.

Synopsis

```
#include <pcs/tis.h>

int fnrange4142(value, mode)
int mode;
double value;
```

Arguments

Item	Range Restrictions/Description
<i>value</i>	Specifies a range code corresponding to the specified range for voltage or current force or measure. Numeric expression Voltage range [V]: -200 to 200 Current range [A]: -1 to 1
<i>mode</i>	Specifies whether <i>value</i> is voltage or current. Integer expression V_RANGE (= 1): voltage range I_RANGE (= 2): current range

Description

This function converts a voltage/current range value to a range code, which stores the DCS control commands in the DCS program memory using the Pcs_hpib_output function.

The relationship between the specified voltage/current value and the range code is shown in the following table:

Mode	Specified Value	Code
Voltage Range	$0 \leq value \leq 2 \text{ V}$	11
	$2 \text{ V} < value \leq 20 \text{ V}$	12
	$20 \text{ V} < value \leq 40 \text{ V}$	13
	$40 \text{ V} < value \leq 100 \text{ V}$	14
	$100 \text{ V} < value \leq 200 \text{ V}$	15
Current Range	$0 \leq value \leq 1 \text{ nA}$	11
	$1 \text{ nA} < value \leq 10 \text{ nA}$	12
	$10 \text{ nA} < value \leq 100 \text{ nA}$	13
	$100 \text{ nA} < value \leq 1 \text{ }\mu\text{A}$	14
	$1 \text{ }\mu\text{A} < value \leq 10 \text{ }\mu\text{A}$	15
	$10 \text{ }\mu\text{A} < value \leq 100 \text{ }\mu\text{A}$	16
	$100 \text{ }\mu\text{A} < value \leq 1 \text{ mA}$	17
	$1 \text{ mA} < value \leq 10 \text{ mA}$	18
	$10 \text{ mA} < value \leq 100 \text{ mA}$	19
	$100 \text{ mA} < value \leq 1 \text{ A}$	20

If you use a VM that is set to differential measurement mode, a voltage range code of 10 ($0 \leq value \leq 0.2 \text{ V}$) is also available. Please note that you cannot use current range code 11 for pulsed spot, pulsed sweep, or sweep with pulsed bias measurements.

If you specify an undefined value, this function returns -1 .

Example

```
. . .
int v_code,i_code;
double volts;
. . .
volts=10.0;
v_code=fnrange(volts,1);
i_code=fnrange(3.e-6,2);
. . .
```

See Also

Fchannel4142
Fnunit4142

Fnunit4142

This function returns a unit address converted from a port address of the DCS.

Synopsis

```
#include <pcs/tis.h>

int fnunit4142(port)
int port;
```

Arguments

Item	Range Restrictions/Description
<i>port</i>	Specifies a DCS port address. Integer expression: 32701 to 32711, 32721 to 32732 ¹

¹ The 32721 to 32732 range is available when a port expander is installed in the SWM.

Description

This function converts a port address to a unit address.

The relationship between the specified port address and the unit address depends on the system configuration.

If an invalid value is specified as the *port* address, or if the specified port is not connected to the DCS terminal, 0 is returned.

Example

```
. . .
int unit, channel;
unit=fnunit4142(32701);
channel=fnchannel4142(fnunit4142(fnsmu(2)));
. . .
```

See Also

Fchannel4142

Pcs_hpib_clear

This function clears a specified HP-IB device.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

int pcs_hpib_clear(eid, addr)
INST eid;
int addr;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies a session ID of the open HP-IB interface.
<i>addr</i>	An integer that specifies the bus address of the device to be cleared.

Description

This function sends a Selected Device Clear (SDC) or a generic Device Clear (DCL) on the HP-IB address. The *eid* specifies the session ID of the open HP-IB interface. The *addr* is the bus address of a device you want to clear. If the *addr* is *-1*, a DCL is sent. If a valid address is specified, an SDC is specified.

Upon any error, *-1* is returned with the error code set to *errno*. Otherwise, *0* is returned.

Example

```
. . .
int ret;
ret=pcs_hpib_clear(eid,7);
. . .
```

See Also

Pcs_hpib_clear

Pcs_hpib_enter

This function reads a string from a specified HP-IB device and returns the length of the string.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

int pcs_hpib_enter(eid, addr, str, n)
INST eid;
int addr, n;
char *str;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies a session ID of the open HP-IB interface.
<i>addr</i>	An integer that specifies the bus address of the device to receive data.
<i>str</i>	Specifies a pointer to a buffer where the data are stored. Any valid pointer.
<i>n</i>	Specifies the size of the data buffer.

Description

This function reads a string from the HP-IB device and the number of bytes of the received string is returned if the string is read successfully. A NULL character ('\0') is added as a read termination character at the end of the character string. If a buffer overflow occurs, the returned length of the string is $n - 1$ because of the NULL character.

The data transfer terminates under one of the following three conditions:

- If a read termination character is received.
- If a character with the EOI line asserted is received.
- If a buffer overflow occurs.

Upon any error, -1 is returned with the error code set to *errno*. Otherwise, 0 is returned.

Example

```
. . .  
INST eid;  
int dvm, len;  
char buff[81];  
dvm=7;  
len=pcs_hpib_enter(eid, dvm, buff, 80);  
. . .
```

See Also

Pcs_hpib_output

Pcs_hpib_open

This function opens an HP-IB interface according to a specified logical unit number and returns the session ID of the HP-IB interface.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

INST pcs_hpib_open(sc)
int sc;
```

Arguments

Item	Range Restrictions/Description
<i>sc</i>	Specifies an HP-IB interface logical unit number.

Description

This function opens the HP-IB interface associated with the logical unit number *sc*. Upon completion, the HP-IB session ID *eid* is returned as the function return value. The returned session ID can then be used as a raw SICL call, to be used for things like checking the bus status.

Upon any error, -1 is returned with the error code set to *errno*.

Example

```
. . .
INST eid;
eid=pcs_hpib_open(8);
. . .
```

See Also

Pcs_hpib_clear

Pcs_hpib_output

This function transfers a string to an HP-IB device.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

int pcs_hpib_output(eid, addr, str, flag)
INST eid;
int addr, flag;
char *str;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies the session ID of the open HP-IB interface.
<i>addr</i>	An integer that specifies the bus address of a device where data is received.
<i>str</i>	Specifies a pointer to a string in which the command characters to be sent are stored. The string must be terminated by a NULL ('\0') character.
	Any valid pointer
<i>flag</i>	An integer that specifies one of the following actions: <i>flag</i> = 0 After sending the string, CR ('\r') and LF ('\n') are sent as write termination characters. <i>flag</i> = 1 The last character of the string is sent with the EOI line asserted. <i>flag</i> = 2 Neither the CR/LF sequence or the EOI assertion is undertaken. A NULL ('\0') character for a string terminator is not sent to the device.

Description

This function outputs commands to a specified HP-IB device. This procedure does not provide any functionality for data formatting. If formatting is necessary, call the C Library Sprintf function to build up the *str*.

Upon any error, -1 is returned with the error code set to *errno*. Otherwise, 0 is returned.

Pcs_hpib_output

Example

```
. . .  
INST eid;  
int ret;  
char command[80];  
strcpy(commands, "F2 R3");  
. . .  
ret=pcs_hpib_output(eid,8,commands,0);  
. . .
```

See Also

Pcs_hpib_clear

Pcs_hpib_spoll

This function performs serial polling on a specified HP-IB device and returns the status byte.

Synopsis

```
#include <pcs/tis.h>
#include <sicl.h>

int pcs_hpib_spoll(eid, addr)
INST eid;
int addr;
```

Arguments

Item	Range Restrictions/Description
<i>eid</i>	Specifies the session ID of the open HP-IB interface.
<i>addr</i>	An integer that specifies the bus address of the serial-pollled device.

Description

This function performs a serial poll on a specified device. It returns an integer in which the lowest byte contains the status byte.

Upon any error, -1 is returned with the error code set to *errno*.

Example

```
. . .
INST eid;
int stat;
. . .
stat=pcs_hpib_spoll(eid,2);
. . .
```

See Also

Pcs_hpib_clear

Pgm_ch_sw_off

These functions store into the DCS program memory the commands that set the output switches of specified unit(s) to off . You can specify one, two, three, or all units with these functions.

Synopsis

```
#include <pcs/tis.h>

int pgm_ch_sw_off(unit)
int unit;

int pgm_ch_sw_off2(unit1, unit2)
int unit1, unit2;

int pgm_ch_sw_off3(unit1, unit2, unit3)
int unit1, unit2, unit3;

int pgm_ch_sw_off_all()
```

Arguments

Item	Range Restrictions/Description
<i>unit</i> to <i>unit3</i>	Specifies a port address or unit address. Port address: Integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port) ¹ Unit address: Integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41421B 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)

¹ The 32721 to 32732 range is available when a port expander is installed in the SWM.

Description

This function stores into program memory the DCS control commands that are equivalent to those of Ch_sw_off. The Pgm_ch_sw_off actually sends the CL . . . command to the DCS.

This function has four variations of calling syntax for specifying one, two, three, or all units.

See the Ch_sw_off reference page for details.

Example

```
. . .  
pgm_start(3);  
    pgm_ch_sw_off(fnsmu(1));  
. . .
```

See Also

Ch_sw_off

Pgm_ch_sw_on

Pgm_ch_sw_on

This function stores into the DCS program memory the commands that set the output switches of specified unit(s) to on. You can specify one, two, three, or all units for this function.

Synopsis

```
#include <pcs/tis.h>

int pgm_ch_sw_on(unit)
int unit;

int pgm_ch_sw_on2(unit1, unit2)
int unit1, unit2;

int pgm_ch_sw_on3(unit1, unit2, unit3)
int unit1, unit2, unit3;

int pgm_ch_sw_on_all()
```

Arguments

Item	Range Restrictions/Description
<i>unit</i> to <i>unit3</i>	Specifies a port address or a unit address. Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port) ¹ Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41421B 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)

¹ The 32721 to 32732 range is available when a port expander is installed in the SWM.

Description

These functions store into program memory the DCS control commands equivalent to Ch_sw_on. Pgm_ch_sw_on actually sends the CN . . . command to the DCS.

This function has four variations of calling synopsis to specify one, two, three, or all units.

See the Ch_sw_on reference page for details.

Example

```
. . .  
pgm_start(3);  
    pgm_ch_sw_on(fnsmu(1));  
. . .
```

See Also

Ch_sw_on
Pgm_ch_sw_off

Pgm_disable_dcs

These functions store into DCS program memory the commands that set specified units to the zero output status.

Synopsis

```
#include <pcs/tis.h>

pgm_disable_dcs(unit)
int unit;

pgm_disable_dcs2(unit1, unit2)
int unit1, unit2;

pgm_disable_dcs3(unit1, unit2, unit3)
int unit1, unit2, unit3;

pgm_disable_dcs_all()
```

Arguments

Item	Range Restrictions/Description
<i>unit1</i> to <i>unit3</i>	Specifies a port address or unit address. Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port) ¹ Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41421B 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)

¹ The 32721 to 32732 range is available when a port expander is installed in the SWM.

Description

These functions store into program memory the DCS control commands that are equivalent to the commands for Disable_port. Disable_port sets specified units to zero output status for the DCS. Pgm_disable_dcs actually sends the DZ . . . command to the DCS.

This function has four variations of the calling synopsis to specify one, two, three, or all units.

Use this function before the end of your DCS function. If the forced value is not set to zero after the DCS program is executed, wet switching at the relays may occur and the pin board relays may be damaged.

See the Disable_port reference page for details.

Example

```
. . .  
pgm_start(3);  
pgm_disable_port2(fnsmu(1), fnsmu(2));  
. . .
```

See Also

Disable_port

Pgm_end

This function is part of the Pgm_start ... Pgm_end context. It terminates the program that stores the control commands into the DCS program memory.

Synopsis

```
#include <pcs/tis.h>

int pgm_end()
```

Description

This function ends the procedure that stores the DCS control commands in the program memory. The procedure is started with the Pgm_start function. This function actually sends the END command to the DCS.

Example

```
pgm_start(2);
pgm_force_v(fnsmu(1), 5.5, 20, 1E-5, 1);
. . .
. . .
pgm_end();
```

See Also

Pgm_start

Pgm_force_i

This function stores into DCS program memory the commands that force a specified current.

Synopsis

```
#include <pcs/tis.h>
```

```
int pgm_force_i(unit, current, range, compliance)
int unit;
double current, range, compliance;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	Specifies a port address or a unit address. Port address: Integer expression 32701 to 32704 (for SMU port) 32729, 32732 (for AUX33 and AUX34 ports of port expander) ¹ Unit address: Integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU)
<i>current</i>	Specifies the output current. Numeric expression [A] For 200 V/1 A SMU: -1 to 1 For 100 V/100 mA SMU: -0.1 to 0.1
<i>range</i>	Specifies the current range of the SMU. Numeric expression [A]: -1 to 1 ² 0 = Auto range
<i>compliance</i>	Specifies the voltage compliance. Numeric expression [V]: -200 to 200 ³

¹ Ports 32729 and 32732 are available when a port expander is installed in the SWM.

² The output *range* depends on the output *current*. For details, see the “Description” of the Force_i reference page.

³ Range restrictions for voltage *compliance* depend on the SMU specified. Refer to the “Description” of the Force_i reference page.

Pgm_force_i

Description

This function stores into the program memory the DCS control commands that set the output current to a specified SMU. This function actually sends the DI . . . command to the DCS.

The arguments of this function are the same as those of the Force_i function. See the Force_i reference page for details.

Example

```
. . .  
pgm_start(1);  
pgm_force_i(fnsmu(1), 1.E-7, 1.E-6, 10.0);  
    pgm_force_i(fnsmu(1),1.E-6);  
. . .
```

See Also

Force_i

Pgm_force_v

This function stores into the DCS program memory the commands that force a specified voltage.

Synopsis

```
#include <pcs/tis.h>

int pgm_force_v(unit, voltage, range, compliance)
int unit;
double voltage, range, compliance;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	Specifies a port address or a unit address. Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port) ¹ Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)
<i>voltage</i>	Specifies the output voltage. Numeric expression [V] For 200 V/1 A SMU: -200 to 200 For 100 V/100 mA SMU: -100 to 100 For VS: -40 to 40
<i>range</i>	Specifies the output range. Numeric expression [V]: -200 to 200 ²
<i>compliance</i>	Specifies the current compliance. Numeric expression [A] For SMUs: -1 to 1 ³ For VSs: — ⁴

¹ Ports 32721 to 32732 are available when a port expander is installed in the SWM.

² The *range* depends on the *voltage*. For details, see the “Description” of the Force_v reference page.

³ Range restrictions for *compliance* depend on the specified *current*. Refer to the “Description” of the Force_i reference page.

⁴ This is dummy argument for VS. The current limiter depends on the output range and is set automatically.

Pgm_force_v

Description

This function stores into program memory the DCS control commands that set the output voltage to a specified SMU or VS. This function actually sends the DV . . . command to the DCS.

The arguments of this function are the same as those of the Force_v function. See the Force_v reference page for details.

Example

```
. . . . .  
pgm_start(1);  
pgm_force_v(fnsmu(1), 3, 20.0, 1.0e-3);  
pgm_force_v(fnsmu(1), 4.0);  
. . . . .
```

See Also

Force_v

Pgm_measure_i

This function stores into the DCS program memory the commands that execute the current measurement of a specified SMU or VS.

Synopsis

```
#include <pcs/tis.h>

int pgm_measure_i(unit, range)
int unit;
double range;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	<p>Specifies a port address or a unit address.</p> <p>Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port)¹</p> <p>Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)</p>
<i>range</i>	<p>Specifies the current range.</p> <p>Numeric expression [A]: -1 to 1 0 for Auto range</p>

¹ Ports 32721 to 32732 are available when a port expander is installed in the SWM.

Description

This function stores into program memory the DCS control commands that measure current at a specified SMU or VS. This function actually sends the TI . . . command to the DCS.

The arguments of this function are not the same as those of the Measure_i function. This function stores the commands that execute the current measurement. To get the measured data and status from the DCS, use the Readout_dcs statement.

See the Measure_i reference page for details.

Pgm_measure_i

Example

```
. . .  
pgm_start(3);  
. . .  
pgm_measure_i(fnsmu(3), 1.E-2);  
. . .
```

See Also

Measure_i
Readout_dcs

Pgm_measure_p

This function stores into the DCS program memory the commands that execute the pulsed spot measurement of a specified SMU or VS.

Synopsis

```
#include <pcs/tis.h>

int pgm_measure_p(unit, mode, range)
int unit, mode;
double range;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	Specifies a port address or a unit address. Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port) ¹ Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU) 32634, 32636, 32638,, 32662, 32664 (for HP 41424A (VS/VMU) VM)
<i>mode</i>	An integer expression that specifies either a voltage or a current measurement. Choose from one of the following macros: V_MEAS (= 1): voltage measurement I_MEAS (= 2): current measurement
<i>range</i>	Specifies the current range or the voltage range. For voltage measurements: Numeric expression [V] 0 for Auto range For SMUs: — ² For VMs: –40 to 40 ³ For current measurements: Numeric expression [A] 0 for Auto range –1 to 1

¹ Ports 32721 to 32732 are available when a port expander is installed in the SWM.

² This argument is ignored for SMUs. The measurement range for the current source of the SMUs is determined by the voltage compliance setting.

³ For grounded measurement mode, the 0 V (Auto range), 2 V, 20 V, and 40 V ranges are available. For differential measurement mode, the 0 V (Auto range), 0.2 V, and 2 V ranges are available.

Pgm_measure_p

Description

This function stores into program memory the DCS control commands that execute a pulsed spot measurement and measure current or voltage at a specified SMU or VM. This function actually sends the **MM3**, **FLO**, and **XE** commands to the DCS. The **RI** . . . or **RV** . . . command is also sent.

The arguments of this function are not the same as those of the **Measure_p** function. This function does not allow you to specify a variable name in which the measured value is returned. The variable name is specified in the **Readout_dcs** statement.

See the **Measure_p** reference page for details.

Example

```
. . .  
pgm_start(1);  
. . .  
    pgm_measure_p(fnsmu(2), I_MEAS, 0.0);  
. . .
```

See Also

Measure_p
Readout_dcs

Pgm_measure_v

This function stores into the DCS program memory the commands that execute the voltage measurement of a specified SMU or VM.

Synopsis

```
#include <pcs/tis.h>

int pgm_measure_v(unit, range)
int unit;
double range;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	<p>Specifies a port address or a unit address.</p> <p>Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port)¹</p> <p>Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU) 32634, 32636, 32638, 32662, 32664 (for HP 41424A (VS/VMU) VM)</p>
<i>range</i>	<p>Specifies the measurement range.</p> <p>Numeric expression [A]: 0 for Auto range For SMUs: —² For VMs: −40 to 40³</p>

1 Ports 32721 to 32732 are available when a port expander is installed in the SWM.

2 This argument is ignored for SMUs. The measurement range for the current source of the SMU is determined by the voltage compliance setting.

3 For grounded measurement mode, 0 V (Auto range), 2 V, 20 V, and 40 V ranges are available. For differential measurement mode, 0 V (auto ranging), 0.2 V, and 2 V ranges are available.

Description

This function stores into program memory the DCS control commands that measure voltage at a specified SMU or VM. This function actually sends the TI . . . command to the DCS.

The arguments of this function are not the same as those of the Measure_v function. This function does not allow you to specify a variable name in which the measured value is returned. The variable name is specified in the Readout_dcs statement.

The *range* is ignored for SMUs because the measurement range is determined by the voltage compliance setting for the current source of the SMUs. If the specified unit is a VM and if the *range* is omitted, the VM uses the Auto range mode.

Pgm_measure_v

See the Measure_v reference page for details.

Example

```
. . .  
pgm_start(1);  
. . .  
    pgm_measure_v(fnsmu(2), 0.0);  
. . .
```

See Also

Measure_v
Readout_dcs

Pgm_search_iv

This function stores into DCS program memory the commands that execute the search measurement.

Synopsis

```
#include <pcs/tis.h>

int pgm_search_iv()
```

Description

This function stores into program memory the DCS control commands that trigger the analog search measurement. This function actually sends the **MM6** and **EX** commands to the DCS.

The analog search conditions are established using the `Pgm_set_asearch` function.

The searched value is returned to the variable specified in the `Readout_dcs` statement. For details, see the `Readout_dcs` reference page.

Example

```
. . .
pgm_start(2);
. . .
pgm_search_iv();
. . .
```

See Also

`Pgm_set_asearch`
`Readout_dcs`

Pgm_set_asearch

This function stores into the DCS program memory the commands that set the arguments for an analog search measurement.

Synopsis

```
#include <pcs/tis.h>

int pgm_set_asearch(srch_unit, sense_unit, srch_mode,
                   meas_mode, start, stop, sense_bias,
                   target, ramp, srch_comp, sense_comp,
                   hold, delay, integ)
int srch_unit, sense_unit, srch_mode, meas_mode;
double start, stop, sense_bias, target, ramp;
double srch_comp, sense_comp, hold, delay, integ;
```

Arguments

Item	Range Restrictions/Description
<i>srch_unit</i>	<p>Specifies a search port address or a search SMU address.</p> <p>Search port address: integer expression 32701 to 32704 (for SMU) 32729, 32732 (for port expander port)¹</p> <p>Search SMU address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU)</p>
<i>sense_unit</i>	<p>Specifies a sense port address or a sense SMU address.</p> <p>Sense port address: integer expression 32701 to 32704 (for SMU) 32729, 32732 (for port expander port)¹</p> <p>Sense SMU address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41420A 200 V/1 A SMU)</p>
<i>srch_mode</i>	<p>An integer expression that specifies the search mode.</p> <p>Choose from the following macros: NEG_FEEDBACK (= 1): negative feedback POS_FEEDBACK (= 2): positive feedback POS_RAMP (= 3): positive ramp NEG_RAMP (= 4): negative ramp</p>
<i>meas_mode</i>	<p>An integer expression that specifies the measurement mode.</p> <p>Choose from the following macros: IS_VM (= -2) IS_VM_MEAS (= -4) IS_IM (= 2) IS_IM_MEAS (= 4) VS_VM (= -1) VS_VM_MEAS (= -3) VS_IM (= 1) VS_IM_MEAS (= 3)</p>
<i>start</i>	<p>Specifies a start value.</p> <p>Numeric expression [V]: -200 to 200²</p>
<i>stop</i>	<p>Specifies a stop value.</p> <p>Numeric expression [V]: -200 to 200²</p>

¹ Ports 32729 and 32732 are available when a port expander is installed in the SWM.

² Actual range depends on the SMU port or SMU used.

Pgm_set_asearch

Item	Range Restrictions/Description
<i>sense_bias</i>	Specifies a sense SMU bias value. Numeric expression If sense SMU is in VS mode [V]: -200 to 200 ¹ If sense SMU is in IS mode [A]: -1 to 1 ¹
<i>target</i>	Specifies the target value. Numeric expression If sense SMU is in VS mode [A]: -1 to 1 ¹ If sense SMU is in IS mode [V]: -200 to 200 ¹
<i>ramp</i>	Specifies the ramp rate. Numeric expression [V/s]: 0.5 to 100000
<i>srch_comp</i>	Specifies the search compliance. Numeric expression [A]: -1 to 1 ¹
<i>sense_comp</i>	Specifies the sense compliance. Numeric expression If sense SMU is in VS mode [V]: -1 to 1 ¹ If sense SMU is in IS mode [A]: -200 to 200 ¹
<i>hold</i>	Specifies the hold time. Numeric expression [s]: 0 to 65.535 Resolution: 0.001
<i>delay</i>	Specifies the delay time. Numeric expression [s]: 0 to 65.535 Resolution: 0.001
<i>integ</i>	Specifies the feedback integration time. Numeric expression [s]: $5E-7$ to 0.45

¹ Actual range depends on the SMU port or SMU used.

Description

This function stores into program memory the DCS control commands that set the arguments for the analog search measurements. This function actually sends the **ASV** . . . , **AIV** . . . or **AVI** . . . , and **AT** . . . commands to the DCS.

The following is a description of the *meas_mode* macros:

Macro	Description
IS_VM	Current measurement at search port; voltage sensing at sense port.
IS_VM_MEAS	Current measurement at search port; voltage sensing at sense port; sensed value returned.
IS_IM	Current measurement at search port; current sensing at sense port.
IS_IM_MEAS	Current measurement at search port; current sensing at sense port; sensed value returned.
VS_VM	Voltage measurement at search port; voltage sensing at sense port.
VS_VM_MEAS	Voltage measurement at search port; voltage sensing at sense port; sensed value returned.
VS_IM	Voltage measurement at search port; current sensing at sense port.
VS_IM_MEAS	Voltage measurement at search port; current sensing at sense port; sensed value returned.

See the Measure_i and the Set_asearch reference pages for more details.

Example

```

. . .
Pgm_set_asearch(fnsmu(2),fnsmu(1),NEG_FEEDBACK,IS_IM_MEAS,0.0,10.0,10.0,3.0e-3,
                100.0,1.0e-3,5.0e-3,0.01,0.001,10e-4);
. . .

```

See Also

Set_asearch
 Pgm_search_iv
 Readout_dcs

Pgm_set_pbias

This function stores into the DCS program memory the commands that set the arguments for the pulsed spot measurements at a specified unit.

Synopsis

```
#include <pcs/tis.h>

int pgm_set_pbias(unit, mode, range, base, peak,
                  width, period, hold, compliance)
int unit, mode;
double range, base, peak, width, period, hold, compliance;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	<p>Specifies a port address or a unit address.</p> <p>Port address: integer expression 32701 to 32704 (for SMU port) 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32732 (for port expander port)¹</p> <p>Unit address: integer expression 32621 to 32628 (for HP 41421B 100 V/100 mA SMU) 32629 to 32632 (for HP 41421B 200 V/1 A SMU) 32633, 32635, 32637, 32661, 32663 (for HP 41424A (VS/VMU) VS)</p>
<i>mode</i>	<p>Integer expression that specifies the output mode.</p> <p>Choose from the following macros: V_SOURCE (= 1): pulsed voltage output I_SOURCE (= 2): pulsed current output</p>
<i>range</i>	<p>A numeric expression that specifies the output range. Choose from the following:</p> <p>If pulsed voltage is forced [V]: -200 to 200^2 If pulsed current is forced [A]: -1 to 1^2</p>
<i>base</i>	<p>A numeric expression that specifies the pulse base.</p> <p>Numeric expression If pulsed voltage is forced [V]: -200 to 200^2 If pulsed current is forced [A]: -1 to 1^2</p>
<i>peak</i>	<p>A numeric expression that specifies the pulse peak.</p> <p>If pulsed voltage is forced [V]: -200 to 200^2 If pulsed current is forced [A]: -1 to 1^2</p>
<i>width</i>	<p>Specifies the pulse width.</p> <p>Numeric expression [s]: 0.001 to 0.05 Resolution: $1\text{E}-4$</p>
<i>period</i>	<p>Specifies the pulse period.</p> <p>Numeric expression [s]: 0, 0.01 to 0.5 Resolution: $1\text{E}-4$</p>
<i>hold</i>	<p>Specifies the hold time.</p> <p>Numeric expression [s]: 0 to 655.35 Resolution: 0.01</p>
<i>compliance</i>	<p>A numeric expression that specifies voltage or current compliance.</p> <p>If pulsed voltage is forced [A]: -1 to 1 If pulsed current is forced [V]: -200 to 200</p>

¹ Ports 32721 to 32732 are available when a port expander is installed in the SWM.

² Actual range depends on the SMU or SMU port used.

Pgm_set_pbias

Description

This function stores into program memory the DCS control commands that set the arguments for the pulsed spot measurements. This function actually sends PV . . . to the DCS for pulsed voltage forcing, or sends PI . . . for pulsed current forcing. PT . . . is sent to set the pulse width, pulse period, and hold time.

See the Set_pbias reference page for more details.

Example

```
. . . . .  
pgm_start(2);  
pgm_set_pbias(32623,I_SOURCE,1E-3,0.0,5E-4,0.1,1.0,0.0,12.0);  
. . . . .
```

See Also

Set_pbias
Pgm_measure_p

Pgm_set_smu

This function stores into the DCS program memory the commands that set the measurement integration time and the filter mode for the SMUs, VSs, and VMs.

Synopsis

```
#include <pcs/tis.h>

int pgm_set_smu(integ_time, filter, samples)
int integ_time, filter, samples;
```

Arguments

Item	Range Restrictions/Description
<i>integ_time</i>	An integer expression that specifies the integration time. Choose from the following: INTEG_MANUAL (= 0): manual INTEG_SHORT (= 1): short INTEG_MEDIUM (= 2): medium INTEG_LONG (= 3): long
<i>filter</i>	An integer expression that specifies the filter mode. Choose from the following: FILTER_OFF (= 0): filter OFF FILTER_ON (= 1): filter ON
<i>samples</i>	An integer expression that specifies the number of samples. 1 to 1023 ¹

¹ This argument is ignored when *integ_time* is set to 1, 2, or 3.

Description

This function stores into program memory the DCS control commands that set the integration time and filter mode for the SMUs, VSs, and VMs of the DCS. This function actually sends AV . . . and FL . . . to the DCS.

See the Set_smu reference page for details.

Example

```
. . .
pgm_start;
pgm_set_smu(INTEG_LONG, FILTER_OFF, 0);
. . .
```

See Also

Set_smu

Pgm_set_vm

This function stores into DCS program memory the commands that set the voltage measurement mode of a specified VM to either grounded measurement mode or differential measurement mode.

Synopsis

```
#include <pcs/tis.h>

int pgm_set_vm(unit, mode)
int unit, mode;
```

Arguments

Item	Range Restrictions/Description
<i>unit</i>	Specifies a VM port address or a VM unit address. VM port address: 32705, 32706, 32708, 32709 (for AUX port) 32721 to 32728, 32730, 32731 (for port expander) ¹ VM unit address: 32634, 32636, 32638, 32662, 32664 (for HP 41424A (VS/VMU) VS)
<i>mode</i>	An integer expression that specifies the measurement mode. Choose from the following macros: VM_GND (= 0): Grounded measurement VM_DIFF (= 1): Differential measurement

¹ Ports 32721 to 32728, 32730, and 32731 are available when a port expander is installed in the SWM.

Description

This function stores into program memory the DCS control commands that set the voltage measurement mode of the specified VM. The voltage measurement mode of the VM is either grounded measurement mode or differential measurement mode. This function actually sends VM . . . to the DCS.

See the Set_vm reference page for details.

Example

```
. . . . .
pgm_start;
Pgm_set_vm(32634, VM_GND)
. . . . .
```

See Also

Set_vm

Pgm_start

This function is part of the Pgm_start ... Pgm_end context. It starts a program that has control commands stored in the DCS program memory. The program is specified by a program number.

Synopsis

```
#include <pcs/tis.h>

int pgm_start(pno)
int pno;
```

Argument

Item	Range Restrictions/Description
<i>pno</i>	An integer expression that specifies a program number. 0 ¹ , 1 to 99

¹ See "Description."

Description

This function starts the procedure that has control commands stored in the DCS program memory. The program is specified by the *pno* argument. This function actually sends the ST · · · command to the DCS.

The program numbers available range from 1 to 99. If 0 is specified as the program number, the DCS control commands sent between the Pgm_start and Pgm_end context are not stored in the program memory, but are immediately executed. For example, the following program lines store DCS control commands that force 10 V and measure current at the SMU1:

```
· · · · ·
pgm_start(1);
    pgm_force_v(fnsmu(1),10.0,20.0,1.0e-3);
    pgm_measure_i(fnsmu(1));
pgm_end();
· · · · ·
```

To execute the DCS control commands stored in the program memory, use Run_dcs_program. To read the measured data, use Readout_dcs or Readsweep_dcs.

See Also

Pgm_end
Run_dcs_program
Readout_dcs
Readsweep_dcs

Pgm_wait

This function stores into program memory the DCS control commands that pause for a specified time.

Synopsis

```
#include <pcs/tis.h>

int pgm_wait(time)
double time;
```

Argument

Item	Range Restrictions/Description
<i>time</i>	Specifies the amount of wait time. Numeric expression [s]: 0 to 99.9999 Resolution: 0.0001

Description

This function stores into program memory the DCS control commands that pause for a specified *time*. This function actually sends the PA . . . command to the DCS.

Example

```
. . .
Pgm_wait(2.0);
. . .
```

Readerror4142

This function gets the error codes that occur between the Pgm_start ... Pgm_end context.

Synopsis

```
#include <pcs/tis.h>

int readerr4142(err1, err2, err3, err4)
int err1, err2, err3, err4;
```

Arguments

Item	Range Restrictions/Description
err1 to err4	Specify error codes from the DCS error code buffer. Any valid integer variable name

Description

Use this function to get error codes that occur between the Pgm_start ... Pgm_end context. This function actually sends the **ERR?** command to the DCS and gets the four error codes from the error register of the DCS. Note that four integer variables must be specified as arguments.

This function is used when you debug the DCS control commands stored in the program memory. The DCS translates the received control commands to the internal codes. The command syntax and its argument contents are checked during the translation. If some errors are detected during the translation, the error codes are stored in the error register of the DCS.

The DCS can hold up to four error codes in the error register and this function can retrieve the errors. If two errors are detected, the two error codes are returned to *err1* and *err2*, and 0 is returned to both *err3* and *err4*. If no errors are detected, 0s are returned to *err1* through *err4*.

For details on the error codes, see the HP 4142B *HP-IB Command Reference Manual*.

Note



Readerror4142 must be executed outside of the Pgm_start ... Pgm_end context; otherwise an error will occur.

You can also get error information during normal execution of TIS by the Error_info function. However, Error_info must also be used outside of the Pgm_start ... Pgm_end context.

Readerror4142

Example

```
main()
{
    INST eid;
    int st;
    int err1, err2, err3, err4;

    eid = pcs_hpib_open(fnaddr4142()/100);

    pgm_start(3);
    pgm_force_v(. . .);
    st = pcs_hpib_spoll(eid, fnaddr4142()%100);
    if ( st & 0x20 ) goto abort;
    . . .
abort:
    pgm_end();
    readerror4142(&err1, &err2, &err3, &err4);
    if ( err1 ) err_rep(err1, err2, err3, err4);
    . . .
}
void err_rep(err1, err2, err3, err4);
int err1, err2, err3, err4;
{
    printf("Error 1:%d\n", err1);
    printf("Error 2:%d\n", err2);
    printf("Error 3:%d\n", err3);
    printf("Error 4:%d\n", err4);
}
```

See Also

Error_info

Readout_dcs

This function requests that the DCS transfer measured data from spot measurements, pulsed spot measurements, or analog search measurements, then returns the measured data to specified variables.

Synopsis

```
#include <pcs/tis.h>

int readout_dcs(data, st)
double *data;
int *st;

int readout_dcs2(data1, st1, data2, st2)
double *data1, *data2;
int *st1, *st2;

int readout_dcs3(data1, st1, data2, st2, data3, st3)
double *data1, *data2, *data3;
int *st1, *st2, *st3;
```

Arguments

Item	Range Restrictions/Description
<i>data</i> to <i>data3</i>	Specifies the measurement value name. Any valid double
<i>st</i> to <i>st3</i>	Specifies the status name. Any valid integer name

Description

This function transfers measured data from the DCS after a program stored in the DCS program memory is executed. This function is used for spot, pulsed spot, and analog search measurements. This function cannot be used between the Pgm_start ... Pgm_end context because the transfer mode is different.

Other than the Pgm_start ... Pgm_end context, the transfer mode is set to the binary mode to reduce the transfer speed. For more information on the binary format, see the HP 4142B *Operation Manual*.

The measurement value name and status name are one pair of data. The following example shows a program that gets one pair of data:

```
. . .
int imeas, stat;
. . .
pgm_start(3);                               Store commands in program memory.
. . .
    pgm_force_v(fnsmu(3),10,20.0,1.0e-3);
    pgm_measure_i(fnsmu(3)0.0);
    pgm_disable_dcs(fnsmu(3));
pgm_end();
```

Readout_dcs

```
. . .  
run_dcs_program(3);           Trigger program in program memory.  
. . .  
readout_dcs(imeas,stat);      Get measured data from DCS.  
. . .
```

The status codes, which may be returned to the *st...st3* arguments, are shown below:

Status	Description of Condition
0	The measurement completed normally.
1	The measurement reached compliance/limit at another unit.
2	The measurement reached compliance/limit at this unit.
3	This unit is oscillating.
4	A measured data overflow is detected.
6 ¹	The target value is not reached during a search (between search start value and search stop value).
7 ¹	The measurement is made before the feedback search is complete.

¹ This status is used only for analog search measurements.

In Offline Mode

Simulated measurement values are returned to *data...data3* and *st...st3*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant data simulation mode:

0.1 is always returned to *data...data3* and 0 is always returned to *st...st3*.

File data simulation mode:

This function reads <meas> and <stat> from the data simulation file and returns these values. The data simulation file format is as follows:

```
Readout_dcs    <meas> <stat> . . . . . <meas> <stat>
```

Use blank spaces, tabs, or commas to delimit values in a data simulation file. The data entries in the file data simulation file must correspond to the arguments of the calling statements.

See Also

Readsweep_dcs
Run_dcs_program

Readsweep_dcs

This function requests that the DCS transfer measured data from sweep measurements. The measured data is returned to specified arrays.

Synopsis

```
#include <pcs/tis.h>

int readsweep_dcs(n, data, st)
double data[];
int n, st[];

int readsweep_dcs2(n, data1, st1, data2, st2)
double data1[], data2[];
int n, st1[], st2[];

int readsweep_dcs3(n, data1, st1, data2, st2, data3, st3)
double data1[], data2[], data3[];
int n, st1[], st2[], st3[];

int readsweep_dcs_m(m, n, data, st)
int m, n, st[m][n];
double data[m][n];
```

Arguments

Item	Range Restrictions/Description
<i>data</i> to <i>data3</i>	Specifies a measurement array name. Any valid double array
<i>st</i> to <i>st3</i>	Specifies a status array name. Any valid integer array.
<i>n</i>	Specifies the number of data to be read. Any valid integer.
<i>m</i>	Specifies the number of units where the sweep measurement is performed. Any valid integer name

Description

This function transfers the measured data from the DCS after the program stored in the DCS program memory is executed. The function is for sweep measurements and cannot be used between the Pgm_start ... Pgm_end context because the transfer mode is different.

Note



The Pgm_#### functions that set sweep arguments are not supported. To set the sweep arguments, you must send the DCS control commands using the Pcs_hpib_output function.

Readsweep_dcs

Other than the Pgm_start ... Pgm_end context, the transfer mode is set to the binary mode to reduce the transfer speed. For more information on the binary format, see the HP 4142B *Operation Manual*.

The measurement array name and status array name are one pair of data. Up to nine pairs of data can be specified using Readsweep_dcs_m. The following example shows a program that gets one pair of data:

```
. . . . .
INST eid;
int  vf_st[31];
double vf[31];
. . . . .
pgm_start(1);                                Store control commands in program memory.
. . . . .
eid=pcs_hpib_open(27);
  pcs_hpib_output(eid,23,"WI3,1,0,0,3E-2,31,4",0);
  pcs_hpib_output(eid,23,"MM2.3",0);
  pcs_hpib_output(eid,23,"XE",0);
  pcs_hpib_output(eid,23,"DZ3",0);
pgm_end();
. . . . .
run_dcs_program(1);                          Trigger program in program memory.
. . . . .
readsweep_dcs(vf,vf_st);                     Get measured data from DCS.
. . . . .
```

The status codes, which may be returned to the *st...st3*, are shown in the following table:

Status	Description of Condition
0	The measurement completed normally.
1	The measurement reached compliance/limit at another unit.
2	The measurement reached compliance/limit at this unit.
3	This unit is oscillating.
4	A measured data overflow is detected.

This function returns -1 if a TIS error is detected. Otherwise, 0 is returned.

In Offline Mode

This function returns simulated measurement values to the *data...data3* and *st...st3*. The type of value returned depends on the simulation mode used: constant data or file data simulation mode.

Constant data simulation mode:

0.1 is always returned to the elements of the *data...data3* and 0 is always returned to the elements of the *st...st3*.

File data simulation mode:

This function reads *<measn>* and *<statn>* from the data simulation file and returns the values. The data simulation file format follows:

Readout_dcs

```
[
<meas1> <stat1> <meas2> <stat2> . . . <meas9> <stat9>      element 1
<meas1> <stat1> <meas2> <stat2> . . . <meas9> <stat9>      element 2
. . . . . . . . . . . . . . . .
<meas1> <stat1> <meas2> <stat2> . . . <meas9> <stat9>      element n
]
```

You can use blank spaces, tabs, or commas to delimit values in a data simulation file. Note that the data entries in the file data simulation file must correspond to the arguments of the calling statements.

See Also

Readout_dcs

Run_dcs_program

Run_dcs_program

This function triggers and executes specified DCS programs stored in the program memory.

Synopsis

```
#include <pcs/tis.h>

int run_dcs_program(pno1)
int pno1;

int run_dcs_program2(pno1, pno2)
int pno1, pno2;

int run_dcs_program3(pno1, pno2, pno3)
int pno1, pno2, pno3;
```

Argument

Item	Range Restrictions/Description
<i>pno1 to pno3</i>	Specify the program number. Integer expression: 1 to 99

Description

These functions trigger a program stored in the DCS program memory. The three functions are for either one, two, or three programs: Run_dcs_program, Run_dcs_program2, or Run_dcs_program3.

Specify the program numbers that are specified by Pgm_start as the arguments of this function. The same program number can be specified more than once.

The DCS programs are executed according to the order of the arguments.

Note



The settings of SMUs, VSs, and VMs are altered when DCS programs are executed, and the settings kept in the TIS memory are different from the actual settings. After executing DCS programs in the program memory, you must use the following functions to return to normal operation of the TIS:

- Init_system
- Disable_port
- Long_cal
- Short_cal

The above functions clear the settings in the TIS memory and reset the DCS.

These functions returns -1 if a TIS error is detected. Otherwise, 0 is returned.

In Offline Mode

This function checks only the arguments.

Example

```
. . .  
run_dcs_program(1);  
. . .  
run_dcs_program(2,4,6,8);  
. . .
```

See Also

Readout_dcs
Readsweep_dcs

Error Messages

Introduction

This section gives a complete listing of the error messages that you might come across while using the C Library. Please see the *Programming Guide for the C Library* for details on how to program using the error handling functions, macros, and variables available in the C Library and the HP 4062UX.

This section lists the error messages for the following HP 4062UX software:

- Pcsstart Errors (Initialization process)
- TIS Errors (C Library Test Instruction Set)
- PCL/FCL Errors (Program Control and File Creation Libraries)
- PGMLIB Errors (DCS Program Library)
- Prober Driver Library Errors

Pcsstart Errors

This section lists errors related to the Pcsstart program of the HP 4062UX. Errors and error messages that occur during Pcsstart are printed automatically.

- pcsstart: Not authorized to use the software.
pcsstart error 0
- pcsstart: Not authorized to use the software (some HP-HIL devices locked).
pcsstart error 0
- pcsstart: Not authorized to use the software. No CODEWORD file found.
pcsstart error 0

The ID module is not connected—or not connected properly—to your system controller, the CODEWORD file does not exist, or the codeword in the CODEWORD file is incorrect.

- pcsstart: SWM/SWC failed power-on test.
pcsstart error 1
- pcsstart: No pin boards installed in SWM.
pcsstart error 2
- pcsstart: SWM down. SRQ asserted.
pcsstart error 3

A hardware failure occurred during the SWC/SWM power-on self-test. Check for one of the following possible causes:

- A momentary power loss may have occurred.

- The control cable interconnecting the SWM and SWC is disconnected, making poor contact, or defective.
- The SWM control bus cable is disconnected, making poor contact, or defective.

After checking the above, rerun the pcsstart program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

- pcsstart: SWM syntax error. SRQ asserted.
pcsstart error 4

The SWC received an illegal program code. Rerun the pcsstart program.

- pcsstart: SWM pin board malfunction. SRQ asserted.
pcsstart error 5

Pin board failure occurred during the SWC/SWM power-on self-test.

- pcsstart: Switching matrix is open.
pcsstart error 6

Enable the SWM by installing the test fixture adapter on the SWM (closes the FIXTURE CLOSED DETECT switch) or by pressing and holding the WAFER PROBER SENSE switch of the SWM, and rerun the pcsstart program.

- pcsstart: This version of switching matrix controller not supported: HP 4084A
REV. *xx.xx*
pcsstart error 7

HP 4062UX supports only the HP 4084B (Rev. 2.0 or later) SWC.

- pcsstart: DCS failed power-on test.
pcsstart error 10

A hardware failure occurred during DCS self-test.

- pcsstart: DCS syntax error. SRQ asserted. : Error *xxx*
pcsstart error 11

The DCS received an illegal program code. Rerun the pcsstart program. The *xxx* above is the DCS error code. Refer to the *HP 4142B Operation Manual*, for details on DCS error codes.

- pcsstart: DCS not interlocked. SRQ asserted.
pcsstart error 12

The interconnect cable assembly is disconnected, making poor contact, or defective. Check these possible causes and rerun the pcsstart program.

- pcsstart: DCS shutdown. SRQ asserted.
pcsstart error 13

DCS output is disabled because of a momentary power loss or because voltage or current in excess of an SMU's maximum current, voltage, or power was applied to an SMU. Rerun the pcsstart program.

- pcsstart: DCS calibration error. SRQ asserted. : Error *xxx* in slot *y*
pcsstart error 14

A hardware failure occurred during DCS calibration. The *xxx* above is the error code of the DCS and *y* is the slot number of the failed module. For example, if the module in slot 1 is not functioning properly, the following error message is displayed:

```
pcsstart: DCS calibration error.  SRQ asserted.   : Error 420 in slot 1
pcsstart error 14
```

If this error occurs, contact your system administrator or nearest HP Service Office. For emergency repair, replace or remove the module in the slot indicated and rerun the pcsstart program.

- pcsstart: CMS syntax error. SRQ asserted.
pcsstart error 15

- pcsstart: CMS illegally programmed. SRQ asserted.
pcsstart error 16

- pcsstart: CMS triggered too fast. SRQ asserted.
pcsstart error 17

Check your cabling configuration and BNC shorting cap connections, and rerun the pcsstart program.

- pcsstart: CMS sweep/bias error. SRQ asserted.
pcsstart error 18

The BNC shorting cap is not connected to the CMS's front panel REMOTE ON/OFF connector. Connect the BNC shorting cap and rerun the pcsstart program.

- pcsstart: CMH terminal improperly connected.
pcsstart error 19

- pcsstart: CMH84 terminal improperly connected.
pcsstart error 19

- pcsstart: CMH terminal connected to CML (AUX*xx*) port on SWM.
pcsstart error 20

- pcsstart: CMH84 terminal connected to CML (AUX*xx*) port on SWM.
pcsstart error 20

- pcsstart: CMH terminal connected to AUX*xx* port on SWM.
pcsstart error 20

- pcsstart: CMH84 terminal connected to AUX*xx* port on SWM.
pcsstart error 20

- pcsstart: CML terminal improperly connected.
pcsstart error 21

- pcsstart: CML84 terminal improperly connected.
pcsstart error 21

- pcsstart: CML terminal connected to CMH (AUX*xx*) port on SWM.
pcsstart error 22

- pcsstart: CML84 terminal connected to CMH (AUX*xx*) port on SWM.
pcsstart error 22

- pcsstart: CML terminal connected to AUX*xx* port on SWM.
pcsstart error 22

- pcsstart: CML84 terminal connected to AUX xx port on SWM.
pcsstart error 22

The CML, CMH, CML84, or CMH84 terminal is connected to the wrong terminal. The xx above is the AUX port number. Check for proper connection between the CMS and SWM, and rerun the pcsstart program.

- pcsstart: CMS INT BIAS switch improperly set.
pcsstart error 23

The INT BIAS switch of the CMS is set to the ± 42 V MAX position. Set the switch to ± 100 V MAX and rerun the pcsstart program.

- pcsstart: CMS cable impedance incorrect.
pcsstart error 24

CMH or CML cable impedance is either too high or too low. Use the cables furnished with the system and rerun the pcsstart program.

- pcsstart: CMU84 SRQ. Error code xxx .
pcsstart error 25

Check the cabling configuration and rerun the pcsstart program.

- pcsstart: Relays for AUX yy of Port Expander can not be controlled.
pcsstart error 26

The port expander relays can not be controlled because one of the port expander relay control boards is either missing or incorrectly installed. The yy above refers to the AUX relays controlled. Check the installation and rerun the pcsstart program.

- pcsstart: HVU/HCU NOT supported (remove HVU/HCU).
pcsstart error 51

This error is reported with a full screen warning message whenever you attempt to install either an HVU or an HCU into the HP 4062UX system. Shared memory space, which is essential to TIS operation, is intentionally destroyed to avoid HVU/HCU operation during TIS programming.

- pcsstart: SYSCONFIG file not found; on-line mode is not available.
pcsstart error 90

Since SYSCONFIG does not exist in the “/usr/pcs/etc” directory, you can perform only offline mode operation.

- pcsstart: The instruments are being used in another session.
pcsstart error 91

Another session already locks the instruments. You can perform only offline mode operation until the session is stopped or changed to offline mode.

- pcsstart: HP-IB timeout is detected at address xxx .
pcsstart error 92

An expected interrupt did not occur within the specified limits. Check for one of the following possible causes:

- The HP-IB cables interconnecting the system controller and instruments were not connected, making poor contact, or defective, when the pcsstart program executed.

A-4 Error Messages

- Instruments were not ON when the pcsstart program was executed.
- HP-IB Interface was not functioning properly.

After checking the above, rerun the pcsstart program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

- pcsstart: DUMCONFIG file (*xxx*) not found.
pcsstart error 100
- pcsstart: Non ASCII character '*xxx*' in DUMCONFIG file.
pcsstart error 101
- pcsstart: Error in DUMCONFIG: File incompletely terminated.
pcsstart error 102
- pcsstart: Error in DUMCONFIG: Confused syntax for 4062 system and for IMA.
pcsstart error 103
- pcsstart: Error in DUMCONFIG: Unrecognized characters included.
pcsstart error 104
- pcsstart: Error in DUMCONFIG: DCS configuration must be described before cabling.
pcsstart error 105
- pcsstart: Error in DUMCONFIG: Invalid pin description:
pcsstart error 110
- pcsstart: Error in DUMCONFIG: Invalid DCS configuration description:
pcsstart error 120
- pcsstart: Error in DUMCONFIG: Invalid slot number:
pcsstart error 121
- pcsstart: Error in DUMCONFIG: *xxx* can not be installed in DCS slot 1.
pcsstart error 122
- pcsstart: Error in DUMCONFIG: *xxx* requires slot X to be empty.
pcsstart error 123
- pcsstart: Error in DUMCONFIG: Slot X is already occupied by *yyy*.
pcsstart error 124
- pcsstart: Error in DUMCONFIG: Two or more units assigned in slot X.
pcsstart error 125
- pcsstart: Error in DUMCONFIG: Unrecognized module *xxxx*.
pcsstart error 126
- pcsstart: Error in DUMCONFIG: Too many units in DCS.
pcsstart error 127
- pcsstart: Error in DUMCONFIG: Invalid cabling description:
pcsstart error 130
- pcsstart: Error in DUMCONFIG: Only Cmu(1) is attachable to CMH port:
pcsstart error 131
- pcsstart: Error in DUMCONFIG: Only Cmu(2) is attachable to CML port:
pcsstart error 132

- pcsstart: Error in DUMCONFIG: Invalid DCS/CMS unit address:
pcsstart error 133
- pcsstart: Error in DUMCONFIG: Invalid SWM port address:
pcsstart error 134
- pcsstart: Error in DUMCONFIG: Corresponding unit not installed:
pcsstart error 135
- pcsstart: Error in DUMCONFIG: Multiple units are attached to SWM port X.
pcsstart error 136
- pcsstart: Error in DUMCONFIG: SMU *xxx* has multiple cabling to SWM ports.
pcsstart error 137
- pcsstart: Error in DUMCONFIG: VS *xxx* has multiple cabling to SWM ports.
pcsstart error 138
- pcsstart: Error in DUMCONFIG: VM *xxx* has multiple cabling to SWM ports.
pcsstart error 139
- pcsstart: Error in DUMCONFIG: Invalid CMS port connection:
pcsstart error 140
- pcsstart: Error in DUMCONFIG: Invalid CMU84 port connection: *xxx*.
pcsstart error 141
- pcsstart: Error in DUMCONFIG: CMU (*xxx*) has multiple cabling to SWM ports.
pcsstart error 142
- pcsstart: Error in DUMCONFIG: CMU84 (*xxx*) has multiple cabling to SWM ports.
pcsstart error 143

The description of DUMCONFIG is not proper. The *xxx* above refers to the unit number. The X refers to a slot or port number. Rerun the pcsstart program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

- pcsstart: Illegal command parameter --- *xxx*.
pcsstart error 201

Rerun the pcsstart program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

- pcsstart: Session *xxx*: Permission denied.
pcsstart error 202

The session that is specified by environment variable PCS_SESSION_ID does not allow accessing. Rerun the pcsstart program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

- pcsstart: Session *xxx* busy.
pcsstart error 203

Another process is making measurements using TIS functions or locks the instruments.

- pcsstart: PCS_SESSION_ID not defined.
pcsstart error 300
- pcsstart: Session ID *xxxx* too long.
pcsstart error 301

A-6 Error Messages

- pcsstart: Group 'pcs' not registered.
pcsstart error 310
- pcsstart: Non ASCII character 'xxx' in SYSCONFIG file.
pcsstart error 320
- pcsstart: Invalid SYSCONFIG entry.
pcsstart error 321
- pcsstart: SYSCONFIG file contains no effective description.
pcsstart error 322
- pcsstart: Error in SYSCONFIG: HP-IB select code must be identical.
pcsstart error 323

HP-IB logical unit number for the HP-IB interface card must be the same in the /usr/pil/etc/hwconfig.cf and /usr/pcs/etc/SYSCONFIG files. Make the same, then perform /usr/pil/bin/pilconf command again. For details about settings of these files, see Chapter 5 in the *HP 4062 Installation Manual*.

- pcsstart: Error in SYSCONFIG: HP-IB address XXX not identical among components.
pcsstart error 324
- pcsstart: Error in SYSCONFIG: Invalid HP-IB address xxx.
pcsstart error 325
- pcsstart: Error in SYSCONFIG: Invalid select code xxx.
pcsstart error 326

Invalid logical unit number xxx. Check the settings in the /usr/pil/etc/hwconfig.cf or /usr/pcs/etc/SYSCONFIG files. Then perform /usr/pil/bin/pilconf command again. For details about settings of these files, see Chapter 5 in the *HP 4062 Installation Manual*.

- pcsstart: Error in SYSCONFIG: xxx is not supported as SWM.
pcsstart error 327
- pcsstart: Error in SYSCONFIG: xxx is not supported as SWC.
pcsstart error 327
- pcsstart: Error in SYSCONFIG: xxx is not supported as DCS.
pcsstart error 327
- pcsstart: Error in SYSCONFIG: xxx is not supported as CMS.
pcsstart error 327
- pcsstart: Device file /dev/rmb/hpibxx not found.
pcsstart error 330
- pcsstart: HP-IB interface card not on select code xxx.
pcsstart error 331

There is not an HP-IB interface card for logical unit number xxx.

- pcsstart: /dev/rmb/hpibxx is not an HP-IB device file.
pcsstart error 332
- pcsstart: HP-IB device file /dev/rmb/hpibxx is not for select code xx.
pcsstart error 333
- pcsstart: /dev/rmb/hpibxx is not a raw HP-IB device file.
pcsstart error 334

- pcsstart: Test session is in bad state; No control terminal.
pcsstart error 340
- pcsstart: Can not create session key file; directory /usr/pcs/sessions not found.
pcsstart error 341
- pcsstart: Error on pcsstart invocation; *xxx*
pcsstart error 350

Check for one of the following possible causes:
 - The system software is not installed properly.
 - System files are destroyed.
 - Cluster node is not added properly.
 Contact your system administrator or nearest HP Service Office.
- pcsstart: Test session pcs_*xxx* is in bad state.
pcsstart error 400
- pcsstart: Test session is in bad state; No shared memory.
pcsstart error 401
- pcsstart: Test session is in bad state; No semaphore.
pcsstart error 402
- pcsstart: Invalid shared memory segment.
pcsstart error 404
- pcsstart: Invalid shared memory segment.
pcsstart error 405
- pcsstart: Invalid semaphore entity.
pcsstart error 406
- pcsstart: Semaphore ID mismatched.
pcsstart error 407
- pcsstart: Session ID inconsistent between key file and shared memory.
pcsstart error 408
- pcsstart: Daemon process ID mismatched.
pcsstart error 409
- pcsstart: Pcsstart co-exists; Shared memory found.
pcsstart error 410
- pcsstart: Pcsstart co-exists; Semaphore found.
pcsstart error 411
- pcsstart: Semaphore removed.
pcsstart error 412
- pcsstart: The online_lock file seems to be snatched by other session.
pcsstart error 413
- pcsstart: Fatal error in pcsstart command.
pcsstart error 450

A-8 Error Messages

Log out and log in again. Then execute the pcsstart command. If the same error is detected, contact your system administrator or nearest HP Service office.

TIS Errors

This section lists errors related to the Test Instruction Set (TIS) of your software. You need to use the error handling facilities of the C Library and the HP 4062UX to see the TIS error numbers and messages.

■ **TIS ERROR 1 Not authorized to use the software.**

The ID module is not connected—or not connected properly—to your system controller. Correct the connection and rerun the pcsstart program.

■ **TIS ERROR 10 HP-IB time out error.**

An expected interrupt did not occur within the specified limits or a system fatal error occurred. Initialize the system by executing the Init_system function. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

■ **TIS ERROR 11 HP-IB SRQ time out error.**

An expected interrupt did not occur within the specified limits. The system's control settings were changed or your HP 4062UX is trying to execute an illegal program operation. Initialize the system by executing the Init_system function.

■ **TIS ERROR 20 Device not present.**

An instrument that was not present during pcsstart (displayed as Not Present) is specified. Check the connection between the instruments and rerun the pcsstart program.

■ **TIS ERROR 21 Invalid port or pin assigned.**

A port or pin board is illegally specified. For example, a VS is specified in a Force_i function; or an illegal pin, such as -5, is specified in a Force_v function.

■ **TIS ERROR 24 Improper pass parameters' value specified.**

Improper pass parameter value in a feedback measurement subprogram.

■ **TIS ERROR 26 Do not execute TIS while using DCS program memory library.**

After invoking a DCS program memory segment with the Run_dcs_program function, you called a regular TIS function. Only an Init_system, Long_cal, Short_cal, or Disable_port function is allowed to be executed after the execution of the DCS program memory segment.

■ **TIS ERROR 30 SWM down. SRQ asserted.**

The SWC or SWM is not functioning properly. Check your cabling configuration and rerun the pcsstart program.

■ **TIS ERROR 31 SWM syntax error. SRQ asserted.**

The SWC received an illegal program code. Initialize your system using the Init_system function.

■ **TIS ERROR 32 SWM open. SRQ asserted.**

SWM output is disabled. This may occur if you're using a wafer prober and your SWM is not mounted properly on the prober; if the test fixture adapter is mounted on SWM but its cover is not closed; or if there is nothing mounted on the SWM.

■ **TIS ERROR 33 Specified pin board not present. SRQ asserted.**

A pin board registered during `pcstart` (pin number displayed) is specified, but cannot be accessed. Check for proper pin board configuration and installation. If you rearranged pin boards, make sure they are properly installed and that the control bus cable is connected.

■ **TIS ERROR 34 Pin board not present.**

A specified pin board is not installed in the SWM. Check for proper pin board configuration and rerun the `pcstart` program.

■ **TIS ERROR 35 (first pin number) >= (last pin number)**

In the calling context of the `Connect_th` function, the first specified pin number is larger than the last specified pin number.

■ **TIS ERROR 40 DCS reports the error code. Error code is *xxxx*.**

An error occurred in the DCS and the DCS reset itself. The *xxxx* is the error code. For the error code of the DCS, see the *HP 4142B HP-IB Command Reference Manual*. To recover from the error, initialize your HP 4062UX using the `Init_system` function.

■ **TIS ERROR 41 DCS not interlocked. SRQ asserted.**

The interconnect cable assembly is disconnected and an attempt was made to output voltage in excess of ± 42 V. Connect the cable assembly.

■ **TIS ERROR 42 DCS shutdown. SRQ asserted.**

The DCS is down. Either the line voltage is down, or a module is seriously damaged and power cannot be supplied to the DCS. Check the line voltage. If the line voltage is correct, contact the nearest HP Service Office.

■ **TIS ERROR 44 VS output voltage incompatible with set voltage range.**

The specified output voltage of a VS exceeds the selected voltage range. Decrease the output voltage or increase the voltage range setting.

■ **TIS ERROR 45 Improper SMU current source, range, or compliance specified.**

The specified compliance value of an SMU is too large for the selected range or specified output value. Decrease the current compliance or the output value, or increase the voltage range setting.

■ **TIS ERROR 46 Illegal sweep power-compliance specified.**

In the calling context of the `Set_iv` function, the specified power compliance is larger than the maximum output power of the SMU. Decrease the power compliance setting.

■ **TIS ERROR 47 Start and stop value are not same polarity.**

In the calling context of the `Set_iv` function, the start value and stop value for a current sweep measurement must be the same polarity.

■ **TIS ERROR 49 Illegal SMU measurement function assigned.**

During a sweep or pulsed measurement, an SMU is set to the wrong mode for the specified measurement. SMUs must be set to VS mode for current measurements, and IS mode for voltage measurements.

■ **TIS ERROR 50 Target value incompatible with compliance value.**

In the calling context of the measurement mode for the `Set_asearch`, `Set_bsearch`, or `Set_lsearch` function, the target value is larger than the compliance value of the sense unit.

- **TIS ERROR 51 This port must be the same V/I mode as the control variable.**

In the calling context of the Set_sync function, the output modes (VS or IS) for the search port and synchronous search port must be the same.

- **TIS ERROR 52 Improper search offset value specified.**

$\text{MAX}(|\text{minimum control value}| \text{ or } |\text{start value}|)$ and $\text{MAX}(|\text{maximum control value}| \text{ or } |\text{stop value}|) + |\text{offset}|$ must be within the search unit's range.

- **TIS ERROR 53 Sense port must be different from search port.**

In the calling context of the Set_asearch function, the search and sense ports are both specified (illegally) as the same port (unit).

- **TIS ERROR 54 Target value out of range.**

The specified target value exceeds the measurement range of the sense unit.

- **TIS ERROR 55 Illegal search v_range, time, and speed combination specified.**

In the calling context of the Set_asearch function, the start value, stop value, ramp rate, and feedback integration time settings are incompatible.

- **TIS ERROR 56 SMU output voltage incompatible with set voltage range.**

The specified output voltage of an SMU exceeds the selected voltage range. Decrease the output voltage or increase the voltage range setting.

- **TIS ERROR 57 Analog Feedback Unit (AFU) not present.**

The AFU is specified for use in a measurement, but is not registered during pcsstart. Make sure your AFU is installed in the DCS and rerun the pcsstart program.

- **TIS ERROR 58 Pulse period incompatible with pulse width.**

The specified pulse width is greater than 50% of the specified pulse period. Decrease the pulse width or increase the pulse period.

- **TIS ERROR 59 Improper DCS sweep, search, and pulse settings.**

In a sweep, search, or pulsed measurement, the measurement execution function executed before you established the measurement conditions. For example, a Sweep_iv function executed before its corresponding Set_iv function executed.

- **TIS ERROR 60 Improper search values (start_value and stop_value) specified.**

In the calling context of the Set_asearch function, the start and stop values must satisfy the proper conditions. See the Set_asearch reference page.

- **TIS ERROR 61 Base_value and peak_value are not same polarity.**

In a pulsed current measurement, the specified pulse base and peak values must be of the same polarity.

- **TIS ERROR 62 Specified VM is still set to Differential Voltage Measurement mode.**

In a pulsed measurement, the differential voltage measurement mode of the VM cannot be used.

- **TIS ERROR 63 Sync range adjustment error.**

The range of the synchronous search port must be the same as the range of the search port. Also, the synchronous search port cannot be set to a range equal to $\text{MAX}(|\text{maximum search value}| \text{ or } |\text{stop value}|)$ and $\text{MAX}(|\text{minimum search value}| \text{ or } |\text{start value}|) + |\text{offset}|$.

- TIS ERROR 64 Cannot specify zero (0) specified for sweep or bias&sweep.

In a pulsed sweep or sweep with pulsed bias measurement, you cannot set the pulse period to 0.

- TIS ERROR 65 Improper pulse measurement pass parameter specified for Set_iv.

In the calling context of the Set_iv function, the specified compliance value is invalid or the logarithmic sweep mode is specified.

- TIS ERROR 66 Improper pulse measurement range specified.

In a pulsed current measurement, the specified current range is too low for the voltage range set. If the voltage range is larger than 2 V, the current range must be $\geq 1\text{E}-4$ A.

- TIS ERROR 67 Total power exceeds the limit. It must be ≤ 32 W.

The total power supplied by the DCS exceeds the maximum rating.

- TIS ERROR 68 Set_sync and Set_iv cannot specify the same port.

The Set_sync and Set_iv functions cannot specify the same port.

- TIS ERROR 69 Set_sync cannot be used with Set_piv or Set_pbias.

The secondary sweep parameter cannot be used with pulsed or pulsed bias sweep measurements.

- TIS ERROR 70 Start or stop value is out of minimum resolution.

Unsuitable start or stop values are set in the Set_iv function, or unsuitable offset or ratio is set in the Set_sync function.

- TIS ERROR 71 VS cannot be used if the primary SMU is current source mode.

When using a VS as a secondary sweep source, the VS and SMU must be the same source mode.

- TIS ERROR 72 Set_sync cannot be used before primary search or sweep setup.

The primary sweep port must be set before the secondary sweep port is set.

- TIS ERROR 73 The secondary sweep port is not set.

The secondary sweep port must be set before the secondary sweep array name is specified.

- TIS ERROR 74 In high voltage state, output relay can't be disconnected.

- TIS ERROR 75 In high voltage state, output relay can't be connected.

DCS cannot disconnect or connect the output relay when the output voltage is more than 42 V.

- TIS ERROR 76 Source SMU or VS unit is masked.

The specified DCS's module is masked by one of the Ch_sw_off functions. Unmask the module by running the pcsstart program or executing the Init_system, Long_cal, or Short_cal functions.

- TIS ERROR 77 TIS cannot be used with this revision.

Quasi-pulsed measurements are available for HP 4142B Rev. 4.0 or later.

- TIS ERROR 80 CMS syntax error. SRQ asserted.

- TIS ERROR 81 CMS illegally programmed. SRQ asserted.

The CMU received an illegal programming code. Initialize your HP 4062UX using the Init_system function.

- TIS ERROR 82 CMS triggered too fast. SRQ asserted.

The CMU received a trigger signal while outputting measurement data from the previous measurement. Make sure that nothing is connected to the EXT TRIGGER terminal of the CMS rear panel.

- TIS ERROR 83 CMS sweep/bias error. SRQ asserted.

The BNC shorting cap was removed from the REMOTE ON/OFF connector of the CMS front panel. Connect the BNC shorting cap and initialize your HP 4062UX using the Init_system function.

- TIS ERROR 84 CMU output voltage incompatible with set voltage range.

The specified output voltage of the CMU exceeds the selected voltage range. Decrease the output voltage or increase the voltage range setting.

- TIS ERROR 85 CMU84 reports an SRQ. Error code is *xxxx*.

The CMU84 caused a service request. The *xxxx* is the error code of the CMU84. For the error code, see the *HP 4284A Operation Manual*.

- TIS ERROR 86 CMU84 Option 001 is not present.

In order to force arbitrary voltage by the HP 4284A (CMU84), the HP 4284A must be furnished with Option 001.

- TIS ERROR 94 Voltage difference between start and stop is less than 10 V.

The conditions for Set_bdv or Set_ileak are not being met.

For the Set_bdv function, the start and stop values must satisfy the following conditions:

$$\square \text{ Stop} \geq \text{start} + 10$$

For the Set_ileak function, the start and voltage values must satisfy the following conditions:

$$\square \text{ Start} \leq \text{voltage} - 10$$

- TIS ERROR 95 Measure_bdv can not be used before Set_bdv.

You called the Measure_bdv function before setting up the measurement conditions with the Set_bdv function.

- TIS ERROR 96 Measure_ileak can not be used before Set_ileak.

You called the Measure_ileak function before setting up the measurement conditions with the Set_ileak function.

- TIS ERROR 100 This TIS function is not supported by off-line mode.

A TIS function that is not supported by offline mode is used.

- TIS ERROR 101 Specified file cannot be opened.

A-14 Error Messages

The file specified by the `Off_line_data` function cannot be opened. One of the following may be the cause:

- ☐ The file does not exist.
- ☐ The directory pathname is incorrect.

■ **TIS ERROR 102 Specified file cannot be opened.**

The file specified by the `Data_creation` function cannot be created. One of the following may be the cause:

- ☐ A file with the same name already exists and is protected.
- ☐ The directory pathname is incorrect.

■ **TIS ERROR 103 Invalid file data is found.**

While reading the data from the file, invalid data are found. Check for one of the following possible causes:

- ☐ TIS function name is not proper: The TIS function that is executed in the program is different from the TIS function that is declared in the file, a name that is not TIS function name is detected, or not allowable characters are used.
- ☐ Measurement data are not proper: data that are not number are detected or not allowable separators are used.
- ☐ Data that is specified in option parameter does not exist in the file.
- ☐ During reading the data, the end of file has come.

■ **TIS ERROR 200 No test session; TIS is used before pcsstart program.**

TIS functions are called before the `pcsstart` program is executed.

■ **TIS ERROR 201 Permission denied to access this session.**

The session that is specified by environment variable `PCS_SESSION_ID` does not allow accessing. Rerun the `pcsstart` program. If the same error code is displayed, contact your system administrator or nearest HP Service Office.

■ **TIS ERROR 202 Session locked.**

Another process in the same session has the instruments locked.

■ **TIS ERROR 203 Test session not ready; TIS in used after pcsstart failed.**

TIS functions are called after the `pcsstart` program is failed.

■ **TIS ERROR 204 Number of session processes exceeds the system limit.**

More than 11 processes in one session call the TIS functions.

■ **TIS ERROR 205 No valid HP-IB address; TIS is used after pcsstart failed.**

TIS functions are called after the `pcsstart` program is failed.

■ **TIS ERROR 220 PCS_SESSION_ID not defined.**

■ **TIS ERROR 221 Session identifier too long.**

■ **TIS ERROR 222 Failed to attach shared memory.**

System software is not installed or customized properly. Contact your system administrator or nearest HP Service Office.

- TIS ERROR 240 Fatal error (errno = *xxx*)
- TIS ERROR 241 Fatal error (errno = *xxx*)
- TIS ERROR 242 Fatal error (errno = *xxx*)
- TIS ERROR 243 Fatal error (errno = *xxx*)
- TIS ERROR 244 Fatal error (errno = *xxx*)
- TIS ERROR 245 Fatal error (errno = *xxx*)
- TIS ERROR 246 Fatal error (errno = *xxx*)
- TIS ERROR 247 Fatal error (errno = *xxx*)
- TIS ERROR 248 Fatal error (errno = *xxx*)
- TIS ERROR 249 Fatal error (errno = *xxx*)
- TIS ERROR 250 Fatal error (errno = *xxx*)
- TIS ERROR 251 Fatal error (errno = *xxx*)
- TIS ERROR 252 Fatal error (errno = *xxx*)
- TIS ERROR 253 Fatal error (errno = *xxx*)
- TIS ERROR 254 Fatal error (errno = *xxx*)
- TIS ERROR 255 Fatal error (errno = *xxx*)
- TIS ERROR 256 Fatal error (errno = *xxx*)
- TIS ERROR 257 Fatal error (errno = *xxx*)
- TIS ERROR 258 Fatal error (errno = *xxx*)
- TIS ERROR 259 Fatal error (errno = *xxx*)
- TIS ERROR 260 Fatal error (errno = *xxx*)
- TIS ERROR 261 Fatal error (errno = *xxx*)
- TIS ERROR 262 Fatal error (errno = *xxx*)
- TIS ERROR 263 Fatal error (errno = *xxx*)
- TIS ERROR 264 Fatal error (errno = *xxx*)
- TIS ERROR 265 Fatal error (errno = *xxx*)
- TIS ERROR 266 Fatal error (errno = *xxx*)
- TIS ERROR 267 Fatal error (errno = *xxx*)
- TIS ERROR 268 Fatal error (errno = *xxx*)
- TIS ERROR 269 Fatal error (errno = *xxx*)

A fatal system error occurred. The *xxx* above is the error code of the system call. Contact your system administrator or nearest HP Service Office.

PCL/FCL Errors

This section lists errors related to the Probing Control and File Creation Libraries. You need to use the error handling facilities of the C Library and the HP 4062UX to see the PCL/FCL error numbers and messages.

■ PCL/FCL ERROR 31 `W_get` is not declared.

The `W_get` function is not included in your measurement program, or does not precede a `W_start_number` or `W_start_xy`.

■ PCL/FCL ERROR 33 `W_start_number` or `W_start_xy` is not declared.

The `W_start_number` or `W_start_xy` function is not included in your measurement program or does not precede a `W_move_next`, `W_move_number`, or `W_move_xy` function.

■ PCL/FCL ERROR 34 Illegal chip coordinate.

The X- and Y-chip coordinates specified in the `W_move_xy` function designate a non-existent chip.

■ PCL/FCL ERROR 35 Chip data is not registered.

A chip not registered during the WAFER configuration stage is specified in a `W_move_number` or `W_move_xy` function. Execute PPG and register the chip, or delete all references to the chip in your `W_move_number` or `W_move_xy` function.

■ PCL/FCL ERROR 36 Module data is not registered.

A module not registered during the CHIP configuration stage is specified in a `W_move_number` or `W_move_xy` function. Execute PPG and register the chip, or delete all references to the chip in your `W_move_number` or `W_move_xy` function.

■ PCL/FCL ERROR 73 Illegal variable name in `W_put_data` or `W_put_array`.

Null data is specified as the string variable in a `W_put_array` or `W_put_data` function.

■ PCL/FCL ERROR 74 Overflow of number of variables.

The number of variables specified in the data file exceeds the limit (50 variables max).

■ PCL/FCL ERROR 81 Specified file cannot be opened.

`W_get`, `W_save_data`, `W_build_file`, or `W_read_file` could not open a specified file.

■ PCL/FCL ERROR 82 Error in probe driver functions.

The probe control library failed due to a probe driver error. The probe driver error message follows this line.

■ PCL/FCL ERROR 83 Duplicate file name (measurement data file).

The data file specified to `W_save_data` or `W_build_data` already exists.

■ PCL/FCL ERROR 84 Number of observations do not match in `w_put_array`.

The number of observations must be equal to each other for each variable that is stored in a file by each `W_save_data` function.

■ PCL/FCL ERROR 85 There are no variables in `W_build_file`.

At least one variable name must be specified to the `W_build_file` parameter list.

PGMLIB Errors

This section lists errors related to the DCS Program Memory Library (PGMLIB). You need to use the error handling facilities of the C Library and the HP 4062UX to see the PGMLIB error numbers and messages.

■ PGMLIB ERROR 1 `Pgm_start` is not declared.

DCS Program Library functions, such `Pgm_force_v`, are used before the `Pgm_start` function.

■ PGMLIB ERROR 2 `Pgm_start` is nested.

`Pgm_start` is called twice before the first `Pgm_start` is ended by a `Pgm_end` function.

■ PGMLIB ERROR 20 Device not present.

The DCS is not online; that is, it is not recognized by the `pcsstart` program.

■ PGMLIB ERROR 21 Invalid port assigned.

An invalid port address or a unit address is specified to the function. Note that a SWM pin number is not accepted as the unit specifier for the PGMLIB functions.

■ PGMLIB ERROR 24 Improper pass parameter specified.

The parameter value passed is incompatible with the corresponding range restrictions.

■ PGMLIB ERROR 100 Error in `Pcs_hpib_open`.

■ PGMLIB ERROR 101 Error in `Pcs_hpib_output`.

■ PGMLIB ERROR 102 Error in `Pcs_hpib_enter`.

■ PGMLIB ERROR 103 Error in `Pcs_hpib_clear`.

■ PGMLIB ERROR 104 Error in `Pcs_hpib_spoll`.

The PGMLIB functions failed to operate on the HP-IB. Check the connection of the HP-IB.

Prober Driver Library Errors

The following is a list of the Prober Driver errors. Note that for the errors to be reported you must use the C prober driver error handling program. You need to use the error handling facilities of the C Library and the HP 4062UX to see prober driver error numbers and messages.

■ PROBER DRIVER ERROR 1 Prober driver is not linked.

A valid prober model specifier is specified for the Prober_init function, but the corresponding prober driver is not linked.

■ PROBER DRIVER ERROR 10 Cannot read SYSCONFIG file.

The Prober_init function failed to open the /usr/pcs/etc/SYSCONFIG file.

■ PROBER DRIVER ERROR 11 Invalid prober type.

An improper prober model specifier is passed to the Prober_init function.

■ PROBER DRIVER ERROR 12 Illegal select code.

The HP-IB interface logical unit number passed to the Prober_init or Prober_identify function is either less than 7 or greater than 31; that is, an inappropriate HP-IB logical unit number.

■ PROBER DRIVER ERROR 13 Illegal bus address.

The HP-IB bus address passed to the Prober_init function is either less than 0 or greater than 30; that is, an inappropriate HP-IB bus address.

■ PROBER DRIVER ERROR 14 Cannot open specified device file.

iopen failed with a specified logical unit number. Confirm that /usr/pil/etc/hwconfig.cf file is set correctly, then perform /usr/pil/bin/pilconf command.

■ PROBER DRIVER ERROR 15 Cannot set/cancel time out.

The prober driver procedure failed to call itimeout(), one of the SICL procedures.

■ PROBER DRIVER ERROR 16 No prober.

The Prober_init function searched the /usr/pcs/etc/SYSCONFIG file to find a prober model corresponding to the specified HP-IB logical unit number and address, but no prober was found.

■ PROBER DRIVER ERROR 17 No prober specified.

No wafer prober is found at the address in which you specified in the Prober_init function.

■ PROBER DRIVER ERROR 18 Cannot define read termination character.

The prober driver procedure failed to call itermchr(), one of the SICL procedures.

■ PROBER DRIVER ERROR 19 Error in Hpib_status_wait or Hpib_bus_status() call.

The prober driver function failed to call igpibbusstatus(), iintroff(), igettimeout(), or iwaitdhr(), which are SICL procedures.

■ PROBER DRIVER ERROR 20 Prober is not online.

The specified prober is not connected or making poor contact.

■ PROBER DRIVER ERROR 21 Miss operation.

Unexpected results caused the prober operation to stop. For example, the prober driver may be in Auto mode and you are attempting to control it locally. Make sure the actual settings match the specified settings.

■ **PROBER DRIVER ERROR 22 Illegal inker ID number.**

A value exceeding the available inker ID numbers was specified.

■ **PROBER DRIVER ERROR 23 Invalid parameter to move.**

An attempt was made to enter a value that exceeds the specified limits.

Index

A

Abbreviations in this manual, 1-3
AFU, 1-3
analog search measurements, 7-5, 7-7
analog search parameters, 7-7
Argument section, 1-2
Assumptions in this manual, 1-3
AUX port names, 12-2
AUX port numbers, 12-2

B

Basic Statistics and Data Manipulation (BSDM)
 system, 14-1
binary search measurements, 7-12
breakdown voltage, 8-2, 8-8

C

Capacitance Measurements, 10-1
capacitance measurements (CMU), 10-2
capacitance measurements (CMU84), 10-4
C·G·t measurements (CMU), 10-23
C·G·V measurements (CMU), 10-27
C·G·V measurements (CMU84), 10-30
Ch_sw_off, 9-2
Ch_sw_off_all, 9-4
Ch_sw_on, 9-6
Ch_sw_on_all, 9-8
C Library Prober Drivers, 13-1
CMH port address, 12-4
CMH terminal address (CMU), 12-6
CMH terminal address (CMU84), 12-7
CML port address, 12-5
CML terminal address (CMU), 12-6
CML terminal address (CMU84), 12-7
CMS, 1-3
CMU, 1-3
CMU84, 1-3
conductance measurements (CMU), 10-2
conductance measurements (CMU84), 10-4
Connect_pin, 4-2
Connect_pin2, 4-2
Connect_pin3, 4-2
Connect_th, 4-5

D

Data_creation, 11-2
data simulation file, 11-2
dc current measurements, 5-19
dc offset voltages, 5-45
DCS, 1-3
DCS channel number, 15-3
DCS device selector, 15-2
DC Source Output Relay Control, 9-1
DCS Program Memory Library, 15-1
DCS Program Memory Library errors, A-18
DCS range code, 15-4
dc voltage measurements, 5-26
Description section, 1-2
differential measurement mode, 5-43
Disable_port, 5-2
Disable_port2, 5-2
Disable_port3, 5-2
Disable_port_all, 5-4
Disconnect_all, 4-7

E

Electroglas 1034X, 13-1
Electroglas 2001X/2010X/3001X/4060X/4080X,
 13-1
Error_info, 11-3
Error Messages, A-1
error message string, 11-3
error number, 11-3
Example section, 1-3

F

FCL Errors, A-17
File Creation Library (FCL), 14-1
Fnaddr4142, 15-2
Fn aux, 12-2
Fnchannel4142, 15-3
Fncmh, 12-4
Fncml, 12-5
Fncmu, 12-6
Fngcmu, 12-8
Fngnd, 12-9
Fngsmu, 12-10
Fnport, 12-11
Fnrange4142, 15-4

Fnsmu, 12-13
Fnunit4142, 15-6
Fnunitsmu, 12-14
Fnvm, 12-15
Fnvs, 12-17
Force_i, 5-5
Force_meas, 5-8
Force_v, 5-12
Function names, 1-2

G

GCMU port (Guard terminal of CMS) address (SWM), 12-8
General Information, 1-1
General I-V Measurements, 5-1
GNDU, 1-3
GNDU port address (SWM), 12-9
grounded measurement mode, 5-43
GSMU port (guard of SMU1) address (SWM), 12-10

H

HP-IB Control, 2-1

I

impedance measurements, 10-4
Init_system, 3-2
Is_on_line, 11-4

L

leakage current, 8-5, 8-11
Limited Auto ranging, 6-38
linear search measurements, 7-16
Lock_system, 3-4
Long_cal, 5-18

M

macros, 1-2
Measure_bdv, 8-2
Measure_cmu, 10-2
Measure_cmu84, 10-4
Measure_i, 5-19
Measure_ileak, 8-4
measurement frequency (CMU84), 10-20
measurement integration time, 5-41
measurement integration time (CMU), 10-8
measurement integration time (CMU84), 10-10
Measure_p, 5-22
Measure_v, 5-26
multichannel staircase sweep measurements, 6-10, 6-47, 6-48
multichannel synchronous staircase sweep measurements, 6-11, 6-47

O

Off_line_data, 11-5
offline data simulation file, 11-2
offline data simulation mode, 11-5
Offline Debugging, 11-1
online test session status, 11-4
Open_corr84, 10-7
OPEN correction (CMU84), 10-7
Open_tis_hpib_fd, 2-2

P

PCL Errors, A-17
Pcs_hpib_clear, 15-7
Pcs_hpib_enter, 15-8
Pcs_hpib_open, 15-10
Pcs_hpib_output, 15-11
Pcs_hpib_spoll, 15-13
Pcsstart, 1-5
Pcsstart Errors, A-1
P_down, 13-3
Pgm_ch_sw_off, 15-14
Pgm_ch_sw_on, 15-16
Pgm_disable_dcs, 15-18
Pgm_end, 15-20
Pgm_force_i, 15-21
Pgm_force_v, 15-23
PGMLIB Errors, A-18
Pgm_measure_i, 15-25
Pgm_measure_p, 15-27
Pgm_measure_v, 15-29
Pgm_search_iv, 15-31
Pgm_set_asearch, 15-32
Pgm_set_pbias, 15-36
Pgm_set_smu, 15-39
Pgm_set_vm, 15-40
Pgm_start, 15-41
Pgm_wait, 15-42
P_home, 13-4
P_imove, 13-7
P_ink, 13-9
P_move, 13-12
P_orig, 13-13
port addresses, 12-11
Port addresses, 12-2
Port and Unit Addresses, 12-1
port names, 12-11
port numbers, 12-11
Port_status, 5-30
power compliance, 6-21
P_pos, 13-15
primary sweep values, 6-11
primary values, 6-39
Prober Driver Library Errors, A-19
Prober_enter, 13-18

- Prober_get_ba, 13-20
- Prober_get_eid, 13-21
- Prober_get_name, 13-22
- Prober_identify, 13-23
- Prober_init, 13-24
- Prober Library file, 13-24
- Prober_open, 13-26
- Prober_output, 13-27
- Prober_read_sysconfig, 13-29
- Prober_reset, 13-30
- Prober_spoll, 13-31
- Prober_status, 13-32
- Prober_wait, 13-34
- Prober_waittime_ctl, 13-35
- Probing Control Library (PCL), 14-1
- Probing Pattern Data file, 14-1
- Probing Pattern Generator (PPG), 14-1
- P_scale, 13-16
- pulsed spot measurements, 5-37
- pulsed sweep measurement conditions, 6-27
- pulsed sweep measurements, 6-2, 6-9, 6-23, 6-36, 6-42, 6-43
- P_up, 13-17

Q

- quasi-pulse, 8-8, 8-11
- Quasi-Pulsed Measurements, 8-1

R

- Readerror4142, 15-43
- Readout_dcs, 15-45
- Readsweep_dcs, 15-47
- real-time multichannel staircase sweep measurements, 6-9
- real-time multiple port sweep measurements, 6-10
- real-time staircase sweep measurements, 6-2
- real time sweep measurements, 6-14
- REMAIN macro, 5-6, 5-10, 5-14, 5-38, 6-21, 6-34, 8-11
- RswEEP_iv, 6-2
- RswEEP_miv, 6-9
- RswEEP_stop, 6-14
- Run_dcs_program, 15-50
- Run-time Error Handling, 11-1

S

- Search_iv, 7-2
- search measurements, 7-2
- Search Measurements, 7-1
- secondary sweep values, 6-11
- secondary values, 6-39
- See Also section, 1-3
- Set_asearch, 7-5

- Set_bdv, 8-7
- Set_bsearch, 7-12
- Set_ct, 10-12
- Set_cv, 10-15
- Set_cv84, 10-18
- Set_freq, 10-20
- Set_ileak, 8-10
- Set_iv, 6-15
- Set_lsearch, 7-16
- Set_pbias, 5-34
- Set_piv, 6-23
- Set_smu, 5-41
- Set_sync (for search measurements), 7-20
- Set_sync (for sweep measurements), 6-30
- settings of the system, 3-2
- Set_tis_hpib_fd, 2-3
- Set_vm, 5-43
- Short_cal, 5-45
- SHORT compensation (CMU84), 10-22
- Short_corr84, 10-22
- SICL, 2-3
- simultaneous real-time sweep measurements, 6-10
- slew rate, 8-8, 8-11
- SMS, 1-3
- SMU, 1-3
- SMU port, 12-13
- SMU unit address, 12-14
- Software Support, 1-4
- spot measurements, 5-22
- staircase sweep measurements, 6-36, 6-38
- staircase sweep with pulsed bias measurements, 6-2, 6-9, 6-36, 6-43
- SWC, 1-3
- Sweep_ct, 10-23
- Sweep_cv, 10-27
- Sweep_cv84, 10-30
- Sweep_iv, 6-36
- sweep measurement conditions, 6-19
- Sweep Measurements, 6-1
- Sweep_miv, 6-47
- sweep parameters, 6-15
- sweep parameters (C·G-t measurements), 10-12
- sweep parameters (C·G-V measurements), 10-15
- sweep parameters for C·G-V measurements (CMU84), 10-18
- sweep port, 6-39
- sweep with pulsed bias measurements, 5-37
- Switching Matrix Control, 4-1
- SWM, 1-3
- synchronous port, 6-30
- synchronous search measurements, 7-20
- synchronous search port, 7-21
- synchronous staircase sweep measurements, 6-9, 6-11, 6-30, 6-36, 6-38

synchronous sweep measurements, 6-5
synchronous sweep port, 6-33, 6-39
synchronous sweep port parameters, 6-33
Synopsis section, 1-2
System Initialization, 3-1
System Locking, 3-1
system settings, 3-2

T

test signal level (CMU), 10-8
test signal level (CMU84), 10-10
TIS, 1-3
Tis_errno, 11-6
TIS error number, 11-3, 11-6
TIS Errors, A-10
Tokyo Seimitsu A-
 PM-3000A/6000A/7000A/60A/80A/90A,
 13-1

U

Unlock_system, 3-6

V

VM unit address, 12-15
VM unit numbers, 12-15
voltage measurement mode, 5-43

voltage sweep measurements, 6-17, 6-32
VS unit address, 12-17
VS unit numbers, 12-17
VS/VMU, 1-3

W

W_build_file, 14-3
W_file_type, 14-5
W_get, 14-6
W_move_next, 14-7
W_move_number, 14-8
W_move_xy, 14-9
W_pos_number, 14-10
W_pos_xy, 14-11
W_put_array, 14-12
W_put_data, 14-14
W_read_file, 14-16
W_return_limit, 14-18
W_return_number, 14-19
W_return_xy, 14-20
W_save_data, 14-21
W_start_number, 14-22
W_start_xy, 14-24
W_total_chip, 14-26