Agilent E5270B
Precision Measurement Solutions

TIS User's Guide

Agilent Technologies

Notices

Manual Part Number

E5270-90040

Edition

Edition 1, October 2004
Edition 2, July 2007
Edition 3, August 2011

Agilent Technologies, Inc.
5301 Stevens Creek Blvd
Santa Clara, CA 95051 USA

Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

In This Manual

The Test Instruction Set (TIS) is a set of powerful functions that facilitate measurement programming for the Agilent E5270B, and supports the following applications:

- DC current/voltage output
- High speed spot measurement
- Multi channel spot measurement
- Pulsed spot measurement
- Staircase sweep measurement
- Pulsed sweep measurement
- Staircase sweep with pulsed bias measurement
- Breakdown voltage measurement
- Leakage current measurement

The Agilent E5270B TIS library is available for the C language users. Measurement programs that include TIS functions are easier to maintain than programs written solely in the C language.

This manual describes the installation and reference information of the Agilent E5270B TIS, and consists of the following chapters:

- "Starting TIS Programming"

  Describes the installation information of the Agilent E5270B TIS library, and general information for programming.

- "TIS Function Reference"

  Describes the reference information of the Agilent E5270B TIS functions.

- "Programming Examples"

  Provides programming examples using the Agilent E5270B TIS functions.

Contents

Contents

Contents

Contents

1 Starting TIS Programming

This chapter describes the installation information of the Agilent E5270B TIS library, and basic information for programming.

- “Installation”
- “TIS Programming”
- “Migration to Agilent 4070”

Installation

This section describes the system requirements and installation procedure for the Agilent E5270B TIS library.

- "System Requirements"
- "To Install TIS Library"

System Requirements

The following system environments are required.

- Operating system

  Microsoft Windows 7 Professional or Windows XP Professional. It must be supported by the application development environment.

- Application development environment (programming environment)

  Microsoft Visual C++ or Borland C++Builder.

- GPIB (IEEE 488) interface and VISA I/O library

  Agilent GPIB interface with Agilent IO Libraries or equivalent.

- Agilent E5270B VXI*plug&play* driver

  The driver is required to use the Agilent E5270B TIS library. The Agilent E5270B TIS library and VXI*plug&play* driver are stored in the Software CD-ROM furnished with the Agilent E5270B.

- Computer and peripherals

  Required specifications depend on the application development environment. See manual of the software.

- Minimum disk space

  4 MB (2 MB for the Agilent TIS library and 2 MB for VXI*plug&play* driver)

NOTE For the latest system requirements, go to www.agilent.com and type in E5270B in the Search field at the top of the page.

To Install TIS Library

The installation flow is shown below. If you have already installed the GPIB (IEEE 488) interface, VISA I/O library, and programming software (Microsoft Visual C++ or Borland C++Builder) on your computer, skip steps 1 through 4.

1. Install the GPIB interface to your PC.

   See manual of the GPIB interface. Note the model number of the GPIB interface, as you may need it to configure the interface (in step 3).

2. Install VISA I/O library.

   Follow the setup program instructions.

3. Configure and check the GPIB interface.

   See manual of the VISA I/O library.

4. Install the programming software.

   Follow the setup program instructions.

5. Install the Agilent E5270B VXI*plug&play* driver.

   a. Insert the Agilent E5270B Series Software CD-ROM to the drive connected to your computer.

   b. Execute \Pnp\E5270.exe on the CD-ROM.

   The setup program installs the driver. See Table 1-1 for the installed files. The TIS library will refer the driver to control the Agilent E5270B.

6. Install the Agilent E5270B TIS library.

   a. Insert the Agilent E5270B Series Software CD-ROM to the drive connected to your computer.

   b. Create a folder (e.g. C:\Agilent\E5270) on your computer.

   c. Copy the Tis subdirectory on the Software CD-ROM to the created folder.

   See Table 1-2 for the installed files.

Table 1-1 Agilent E5270B VXIplug&play Driver Files

File Name [a]	Description
<install folder>\Winnt\Age5270\age5270.bas	Driver for Microsoft Visual Basic
<install folder>\Winnt\Age5270\age5270.c	Driver source code
<install folder>\Winnt\Age5270\age5270.def	DLL export definition file
<install folder>\Winnt\Age5270\age5270.fp	Front panel file
<install folder>\Winnt\Age5270\age5270.h	Driver header file
<install folder>\Winnt\Age5270\age5270.hlp	On-line help file
<install folder>\Winnt\Age5270\readme.txt	Read me file
<install folder>\Winnt\bin\age5270_32.dll	Driver DLL file
<install folder>\Winnt\include\age5270.h	Driver header file
<install folder>\Winnt\lib\bc\age5270.lib	Library for Borland C++Builder
<install folder>\Winnt\lib\bc\age5270_32.lib	Library for Borland C++Builder
<install folder>\Winnt\lib\msc\age5270.lib	Library for Microsoft Visual C++
<install folder>\Winnt\lib\msc\age5270_32.lib	Library for Microsoft Visual C++

a. Execute `echo %VXIPNPPATH%` on the Command Prompt to know <install folder>.

Table 1-2 **Agilent E5270B TIS Library Files**

File Name [a]	Description
<user path>\lib\msc\E5270_TIS.lib	TIS library for Microsoft Visual C++
<user path>\lib\bc\E5270_TIS.lib	TIS library for Borland C++Builder
<user path>\E5270_TIS.h	TIS library header file
<user path>\E5270_TIS.c	TIS library source code
<user path>\Tis_gd.pdf	TIS User's Guide
<user path>\sample\sample.c	Sample program source code. This is just example program.
<user path>\sample\makefile	Sample program makefile for Microsoft Visual C++

a. <user path> indicates the folder created when you install the TIS library (e.g. C:\Agilent\E5270).

TIS Programming

This section provides the basic information of the Agilent E5270B TIS programming.

- “To Create Your Project Template”
- “To Create Measurement Program”
- “Debugging”

To Create Your Project Template

Before starting the E5270B TIS programming, create your project template, and keep it as your reference. It will remove the conventional task in the future programming.

Step 1. Launch the programming software and create a new project. Then, select the Win32 project or the console application for the new project template selection. They will simplify the programming. Of course, other project template can be used.

Step 2. Define the followings to the project properties or the project options. See manual or on-line help of the programming software for defining them.

1. Additional include file search path:
   - directory (e.g. \Agilent\E5270) that stores the E5270_TIS.h file
   - directory (e.g. \Program Files\VISA\winnt\include) that stores the age5270.h file and the VISA related include files
2. Additional library search path:
   - directory (e.g. \Agilent\E5270\lib\msc for Microsoft Visual C++ or \Agilent\E5270\lib\bc for Borland C++Builder) that stores the E5270_TIS.lib file
   - directory (e.g. \Program Files\VISA\winnt\lib\msc for Microsoft Visual C++ or \Program Files\VISA\winnt\lib\bc for Borland C++Builder) that stores the age5270.lib file and the VISA related library files
3. Additional project link library:
   - E5270_TIS.lib
   - age5270.lib

Step 3. Open a source file (.cpp) in the project, and enter a program code as template. See Table 1-3 for example. The example uses Microsoft Visual C++.

Step 4. Save the project as your template (e.g. \test\my_temp).

Table 1-3 Example Template Program Code for Visual C++

```
#include <stdio.h>                                                          /* 1 */
#include "E5270_TIS.h"

main()
{
 int ret;
 #ifdef E5270_TIS_H                                                         /* 7 */
    ret = open_E5270("GPIB0::17::INSTR", ERR_DETECT_ON, NULL);
    if ( ret != 0 ) {
                     printf( "Failed E527x connection. Exits program.\n" );
                     return( -1 );
                    }
 #endif                                                                     /* 13 */

 /* Insert measurement program code here. */

 #ifdef E5270_TIS_H                                                         /* 17 */
    close_E5270();
 #endif                                                                     /* 19 */
 return( 0 );
}
```

Line	Description
1 and 2	Required to use the Agilent E5270B TIS library. The header files contain various necessary information such as function declaration and macro definitions. You may add the include statements to call another header files that may be needed by the codes you added. Also, the include statements may be written in a header file that will be called by the source file (e.g. #include <stdio.h> may be written in the stdafx.h header file that will be called by the source file).
8	Establishes the software connection with the Agilent E5270B. The above example is for the Agilent E5270B on the GPIB address 17. Confirm the GPIB address of your E5270B, and set the address correctly instead of “17”.
18	Disables the software connection with the Agilent E5270B.

7 to 13
 	
17 to 19
	The E5270_TIS_H is a macro that is effective after the E5270_TIS.h file is included. The macro can be used to judge whether the E5270B TIS library is available or not, and switch a program operation as the following example.
	
	`#ifdef E5270_TIS_H`
	`  /* codes for E5270B */`
	`#else`
	`  /* codes for 4070 */`
	`#endif`
	
	This is used to switch codes to be performed. If the E5270_TIS_H is effective, the codes for E5270B will be performed. Else, the codes for 4070 will be performed.

To Create Measurement Program

Create the measurement program as shown below. The following procedure needs your project template. If the procedure does not fit your programming environment, arrange it to suit your environment.

Step 1. Plan the automatic measurements. Then decide the following items:

- Measurement devices

  Discrete, packaged, on-wafer, and so on.

- Parameters/characteristics to be measured

  h_{FE}, Vth, sheet resistance, and so on.

- Measurement method

  Spot measurement, staircase sweep measurement, and so on.

Step 2. Make a copy of your project template (e.g. \test\my_temp to \test\dev_a\my_temp).

Step 3. Rename the copy (e.g. \test\dev_a\my_temp to \test\dev_a\spot_id).

Step 4. Launch the programming software.

Step 5. Open the project (e.g. \test\dev_a\spot_id).

Step 6. Open the source file that contains the template code as shown in Table 1-3, and insert your measurement program code at the appropriate position. Then use the E5270B TIS functions:

- init_system: to initialize the Agilent E5270B
- connect_pin: to connect device under test
- force_i, force_v, etc.: to set source outputs
- measure_v, measure_i, etc.: to perform measurements
- disable_port: to disable source/measurement channels
- disconnect_all: to disconnect device under test

Step 7. Insert the code for displaying data, storing data, or calculating data.

Step 8. Save the project (e.g. \test\dev_a\spot_id).

NOTE For the example programs, see Chapter 3, “Programming Examples”.

Debugging

The port_status function returns measurement status of specified port. The following table shows all statuses returned by port_status or status_miv function.

Unit	Returned Value	Condition
SMU	NORMAL_MEAS (= 0)	Normal
	DCS_COMP_OTHER (= 1)	Another unit has reached compliance
	DCS_COMP (= 2)	This unit has reached compliance
	DCS_OSC (= 3)	This unit is oscillating.
	DCS_OVERFLOW (= 4)	Measurement data exceeds the measurement range.
	DCS_SWP_STOPPED (= 5)	Sweep was aborted by compliance condition. (stop mode = 2 in set_iv or set_piv)
GNDU	NORMAL_MEAS (= 0)	Normal

NOTE

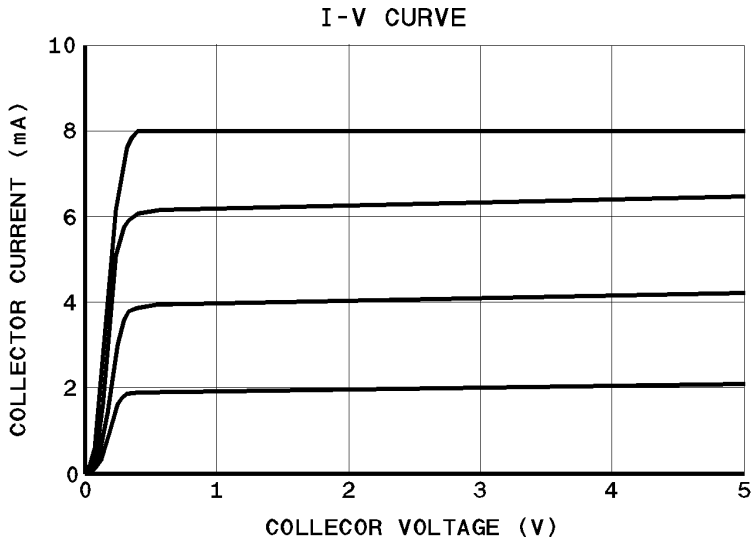
If the port_status function is executed for a measurement port that just finished a sweep measurement (sweep_iv or sweep_miv), only the status for last sweep step is returned.

Monitoring if SMU Reaches Compliance

You can use the port_status function to monitor whether an SMU reached saturation, that is, the compliance value set by force_i, force_v, set_iv, and so on.

Figure 1-1 shows an example in which the compliance value was set to 8 mA for the SMU. As shown in the graph, the top curve is flat and does not increase because the compliance is set to 8 mA.

Figure 1-1 **Reaching Compliance (8 mA): Ic-Vce Characteristics**

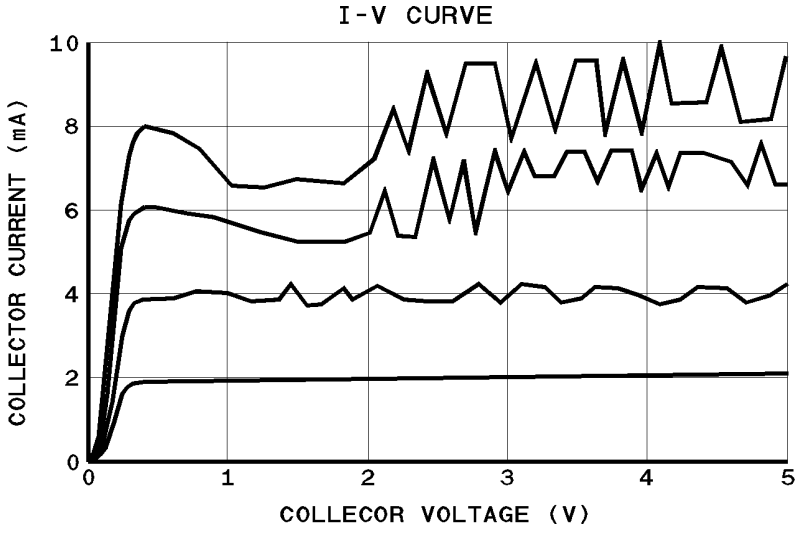


Monitoring if SMU Oscillates

You can use the port_status function to monitor if an SMU oscillated.

Figure 1-2 shows an example graph for oscillation.

Figure 1-2 **Oscillation: Ic-Vce Characteristics**



Migration to Agilent 4070

The Agilent E5270B TIS library is a set of functions that provide the same synopsis as the Agilent 4070 TIS functions. So, if you create programs using the Agilent E5270B TIS library, the programs can be used to control the Agilent 4070 tester with small change.

To migrate from the Agilent E5270B to Agilent 4070 successfully, perform the following procedure.

1. Open the program on the Agilent 4070 system controller (HP-UX computer).
2. Change the header file. The header file `E5270_TIS.h` is not required. Include the `tis.h` file instead.

   ```
   #include "/opt/hp4070/include/tis.h"
   ```

3. Remove the following functions. They are not required for the Agilent 4070.
   - open_E5270
   - close_E5270
4. Remove the VXI*plug&play* driver functions for Agilent E5270B if they are used in the program. The driver cannot be used to control the Agilent 4070. This part of program should be created and added to the program for the Agilent 4070.
5. Remove the `vi_E5270` function if it is used in the program. The function cannot be used to control the Agilent 4070. This part of program should be created and added to the program for the Agilent 4070.
6. Refer to Table 1-4 and correct differences between the E5270B TIS and 4070 TIS.

NOTE

E5270_TIS_H macro

If the program includes the E5270_TIS_H macro as shown below, this part of program does not have to be deleted.

```
#ifdef E5270_TIS_H
   /* codes for E5270B */
#else
   /* codes for 4070 */
#endif
```

Codes for 4070 will be performed if the E5270_TIS_H macro is not effective.

Table 1-4 **Differences between E5270B TIS and 4070 TIS**

Function or Item	Explanation
connect_pin	The *pin* value must be a pin number actually available for the switching matrix of the 4070.
set_adc	The parameter values are compatible. But the actual setup value is different for INTEG_SHORT.
set_bdv and set_ileak	Followings are different between the E5270B and 4070. • Maximum *delay* (65.535 s for 4070, 6.5535 s for E5270B)
set_pbias and set_piv	Followings are different between the E5270B and 4070. • Restrictions of setting *period* and *width* values • *period* and *width* values automatically set when you specify *period*=0
set_piv	Logarithmic sweep is available for 4070.
set_sync	The *offset* value is effective for logarithmic sweep using 4070.
set_timestamp	For 4070, this function must be executed before the port_status or status_miv function.
sweep_iv and sweep_miv	Returned data when a sweep abort condition occurred is different as shown in Table 1-5 and Table 1-6. The tables show an example that an abort condition occurs at step=N.
Current range value	200 mA range is not supported by the 4070. Use 1 A range of HPSMU instead. For 4070, 100 pA range is also available.
Initial settings	They are not the same because of the differences of hardware. If your program includes error handling part, check the error code, and correct the program.
Error codes and messages	

Table 1-5 E5270B returned data when a sweep abort condition occurred

step	*measure* or *meas_vals*	*source*	*sync*	*statuses*	*times*
1	(measured data)	(output data)	(output data)	0	(time data)
:	:	:	:	:	:
N-1	(measured data)	(output data)	(output data)	0	(time data)
N	(measured data)	(output data)	(output data)	X[a]	(time data)
N+1	9999999.99999	0	0	5	0
:	:	:	:	:	:
Last	9999999.99999	0	0	5	0

a. X will be a status code that indicates a sweep abort condition.

Table 1-6 4070 returned data when a sweep abort condition occurred

step	*measure* or *meas_vals*	*source*	*sync*	*statuses*	*times*
1	(measured data)	(output data)	(output data)	0	(time data)
:	:	:	:	:	:
N-1	(measured data)	(output data)	(output data)	0	(time data)
N	9999999.99999	(output data)	(output data)	X[a]	(time data)
N+1	(no data)	(no data)	(no data)	5	0
:	:	:	:	:	:
Last	(no data)	(no data)	(no data)	5	0

a. X will be a status code that indicates a sweep abort condition.

2 TIS Function Reference

This chapter is the complete reference of the Agilent E5270B TIS functions.

- “How to Use the Reference Pages”
- “Function Reference”

How to Use the Reference Pages

Each description module for a function consists of the following sections:

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

The description module will not contain the section that does not apply to the function.

Function Name

The function name is shown at the top of the description module with a brief description.

Synopsis

The synopsis of the function is printed in `computer` font as shown in the following example:

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

int port;
double voltage,range,compliance;
```

The first line shows the function type and arguments of the function, and remaining lines show type of each argument.

Arguments

The arguments of a function specify values to pass to the function or where to return values to a function.

This section briefly lists and describes each argument in a table. The details about each argument are described in the Description section. The arguments are indicated by *italic* type in the text.

Several pre-defined macros are provided that can be used for some function arguments. The allowed macros for each function are listed. The macros are UPPERCASE, so the macro must be UPPERCASE letters when used in a program.

NOTE

Return value

The functions of which the return value is not described exit with one of the following value:

0:	Successful completion
-1:	Error occurred

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` font.

See Also

The See Also section lists the related functions.

Function Reference

This section describes the Agilent E5270B TIS functions. The functions are listed in alphabetical order.

Function List

Table 2-1 summarizes the Agilent E5270B TIS functions.

Table 2-1 Agilent E5270B TIS Function Summary

Category	Function	Summary
Starting	open_E5270	Establishes the software connection to the Agilent E5270B.
Ending	close_E5270	Terminates the software connection to the Agilent E5270B.
Reset	init_system	Initializes the Agilent E5270B.
Query	error_info	Returns a TIS error number and an error message string.
	port_status, port_status_t	Returns the status of the specified port and time stamp.
	status_miv	Returns the status of the data measured by sweep_iv or sweep_miv function, and time stamp.
Channel/Port control	connect_pin	Enables source/measurement channel.
	disconnect_all	Disables all channels.
	set_smu_ch	Selects A/D converter type and filter ON or OFF for the specified SMU.
	disable_port	Sets the specified channel to the zero output state.
A/D converter setup	set_adc	Sets the operation parameters of A/D converter.
Time stamp setup	reset_timestamp	Enables to use the force_i or force_v function to clear the time stamp.
	set_timestamp	Enables or disables the time stamp data output.

Category	Function	Summary
Source output	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
High speed spot measurement	measure_i, measure_it	Measures DC current and returns the measurement data and time stamp.
	measure_v, measure_vt	Measures DC voltage and returns the measurement data and time stamp.
Multi channel spot measurement	measure_m	Performs multi channel DC voltage or current measurements using up to eight SMU channels.
Pulsed spot measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_pbias	Sets the pulse source parameters.
	measure_p	Performs a pulsed spot measurement and returns the measurement value.
Staircase sweep measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_iv	Sets the setup parameters for staircase sweep measurements.
	set_sync	Sets the synchronous sweep source.
	sweep_iv	Performs sweep measurement and returns measurement data.
Staircase sweep measurement using multiple measurement channels	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_iv	Sets the setup parameters for staircase sweep measurements.
	set_sync	Sets the synchronous sweep source.
	sweep_miv	Performs sweep measurements using the up to 8 measurement channels and returns measurement data.

Category	Function	Summary
Pulsed sweep measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_piv	Sets the setup parameters for pulsed sweep measurements.
	set_sync	Sets the synchronous sweep source.
	sweep_iv	Performs pulsed sweep measurement and returns measurement data.
Staircase sweep with pulsed bias measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_iv	Sets the setup parameters for staircase sweep measurements.
	set_pbias	Sets the pulse source parameters.
	set_sync	Sets the synchronous sweep source.
	sweep_iv	Performs sweep measurement and returns measurement data.
Breakdown voltage measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_bdv	Sets the setup parameters for breakdown voltage measurement.
	measure_bdv	Performs breakdown voltage measurement and returns measurement data.
Leakage current measurement	force_i	Forces current from the specified SMU.
	force_v	Forces voltage from the specified SMU.
	set_ileak	Sets the setup parameters for leakage current measurement.
	measure_ileak	Performs leakage current measurement and returns measurement data.

Arguments

The arguments used by several functions are explained in this section.

- “Port address”
- “range value and ranging mode”
- “Output voltage, resolution, and compliance by range”
- “Output current, resolution, and compliance by range”

Table 2-2 **Port address**

Macro	Value [a]	Description
SMU1	20001	Specifies SMU installed in the Agilent E5270B. SMU1 always specifies the SMU installed in a slot closest to the slot 1. SMU2 and followings specify SMU following to the previous one. where slot 1 is the left top slot, and slot 8 is the right bottom slot. Available port addresses depend on the number of SMUs installed in the Agilent E5270B.
SMU2	20002	
SMU3	20003	
SMU4	20004	
SMU5	20005	
SMU6	20006	
SMU7	20007	
SMU8	20008	
GNDU	20009	Specifies the ground unit. Available for the connect_pin, port_status, and port_status_t functions.

a. For program readability and future compatibility, it is recommended not to use this value directly but use macro instead.

Table 2-3 ***range*** **value and ranging mode**

Voltage or current	Available *range* values [a] [b]	Ranging mode used for output/measurement
both	*range* = 0	Auto ranging
voltage for HPSMU	0 < |*range*| ≤ 2 V	2 V limited auto ranging
	2 V < |*range*| ≤ 20 V	20 V limited auto ranging
	20 V < |*range*| ≤ 40 V	40 V limited auto ranging
	40 V < |*range*| ≤ 100 V	100 V limited auto ranging
	100 V < |*range*| ≤ 200 V	200 V limited auto ranging
voltage for MPSMU/HRSMU	0 < |*range*| ≤ 0.5 V	0.5 V limited auto ranging
	0.5 < |*range*| ≤ 2 V	2 V limited auto ranging
	2 V < |*range*| ≤ 5 V	5 V limited auto ranging
	5 V < |*range*| ≤ 20 V	20 V limited auto ranging
	20 V < |*range*| ≤ 40 V	40 V limited auto ranging
	40 V < |*range*| ≤ 100 V	100 V limited auto ranging
current for HPSMU	0 < |*range*| ≤ 1 nA	1 nA limited auto ranging
	1 nA < |*range*| ≤ 10 nA	10 nA limited auto ranging
	10 nA < |*range*| ≤ 100 nA	100 nA limited auto ranging
	100 nA < |*range*| ≤ 1 µA	1 µA limited auto ranging
	1 µA < |*range*| ≤ 10 µA	10 µA limited auto ranging
	10 µA < |*range*| ≤ 100 µA	100 µA limited auto ranging
	100 µA < |*range*| ≤ 1 mA	1 mA limited auto ranging
	1 mA < |*range*| ≤ 10 mA	10 mA limited auto ranging
	10 mA < |*range*| ≤ 100 mA	100 mA limited auto ranging
	100 mA < |*range*| ≤ 1 A	1 A limited auto ranging

Voltage or current	Available *range* values [a] [b]	Ranging mode used for output/measurement
current for MPSMU	0 < |*range*| ≤ 1 nA	1 nA limited auto ranging
	1 nA < |*range*| ≤ 10 nA	10 nA limited auto ranging
	10 nA < |*range*| ≤ 100 nA	100 nA limited auto ranging
	100 nA < |*range*| ≤ 1 µA	1 µA limited auto ranging
	1 µA < |*range*| ≤ 10 µA	10 µA limited auto ranging
	10 µA < |*range*| ≤ 100 µA	100 µA limited auto ranging
	100 µA < |*range*| ≤ 1 mA	1 mA limited auto ranging
	1 mA < |*range*| ≤ 10 mA	10 mA limited auto ranging
	10 mA < |*range*| ≤ 100 mA	100 mA limited auto ranging
current for HRSMU	0 < |*range*| ≤ 1 pA (with ASU)	1 pA limited auto ranging
	1 pA < |*range*| ≤ 10 pA (with ASU)	10 pA limited auto ranging
	0 < |*range*| ≤ 10 pA (without ASU)	10 pA limited auto ranging
	10 pA < |*range*| ≤ 100 pA	100 pA limited auto ranging
	100 pA < |*range*| ≤ 1 nA	1 nA limited auto ranging
	1 nA < |*range*| ≤ 10 nA	10 nA limited auto ranging
	10 nA < |*range*| ≤ 100 nA	100 nA limited auto ranging
	100 nA < |*range*| ≤ 1 µA	1 µA limited auto ranging
	1 µA < |*range*| ≤ 10 µA	10 µA limited auto ranging
	10 µA < |*range*| ≤ 100 µA	100 µA limited auto ranging
	100 µA < |*range*| ≤ 1 mA	1 mA limited auto ranging
	1 mA < |*range*| ≤ 10 mA	10 mA limited auto ranging
	10 mA < |*range*| ≤ 100 mA	100 mA limited auto ranging

a. For the functions to start or execute measurement, negative *range* values are available. The negative values set the ranging mode to the fix, not the limited auto.
b. For the functions to start or execute the measurement that uses the pulse source, set 0 or positive value to set the minimum range that covers the compliance value automatically.

NOTE Auto ranging mode

SMU uses the optimum range to force/measure voltage or current.

NOTE Limited auto ranging mode

SMU uses the optimum range to force/measure voltage or current. Then, the SMU never uses the range less than the specified range.

Table 2-4 **Output voltage, resolution, and compliance by range**

Output range (actually used)	Setting resolution in V	Output voltage [a] in V	Maximum *comp* value [b] in A		
			HPSMU	MPSMU	HRSMU
0.5 V	25E-6	0 to ± 0.5	NA	±100E-3	±100E-3
2 V	100E-6	0 to ± 2	±1	±100E-3	±100E-3
5 V	250E-6	0 to ± 5	NA	±100E-3	±100E-3
20 V	1E-3	0 to ± 20	±1	±100E-3	±100E-3
40 V	2E-3	0 to ± 20	±500E-3	±100E-3	±100E-3
		to ± 40		±50E-3	±50E-3
100 V	5E-3	0 to ± 20	±125E-3	±100E-3	±100E-3
		to ± 40		±50E-3	±50E-3
		to ± 100		±20E-3	±20E-3
200 V	10E-3	0 to ± 200	±50E-3	NA	NA

a. Parameter name may be *base*, *bias*, *peak*, *value*, *start*, *stop*, and so on.
b. This column shows the maximum value of the current compliance.

Table 2-5 **Output current, resolution, and compliance by range**

Output range (actually used)	Setting resolution in A	Output current [a] in A	Maximum *comp* value [b] in V		
			HPSMU	MPSMU	HRSMU
1 pA	1E-15	0 to ± 1.15 E-12	NA	NA	±100
10 pA	5E-15	0 to ± 11.5 E-12			±100
100 pA	5E-15	0 to ± 115 E-12			±100
1 nA	50E-15	0 to ± 1.15 E-9	±200	±100	±100
10 nA	500E-15	0 to ± 11.5 E-9	±200	±100	±100
100 nA	5E-12	0 to ± 115 E-9	±200	±100	±100
1 µA	50E-12	0 to ± 1.15E-6	±200	±100	±100
10 µA	500E-12	0 to ± 11.5E-6	±200	±100	±100
100 µA	5E-9	0 to ± 115E-6	±200	±100	±100
1 mA	50E-9	0 to ± 1.15E-3	±200	±100	±100
10 mA	500E-9	0 to ± 11.5E-3	±200	±100	±100
100 mA	5E-6	0 to ± 20E-3	±200	±100	±100
		to ± 50E-3	±200	±40	±40
		to ± 100E-3	±100	±20	±20
		to ± 115E-3	±100	NA	NA
1 A	50E-6	0 to ± 50E-3	±200		
		to ± 125E-3	±100		
		to ± 500E-3	±40		
		to ± 1	±20		

a. Parameter name may be *base*, *bias*, *peak*, *value*, *start*, *stop*, and so on.
b. This column shows the maximum value of the voltage compliance.

close_E5270

This function terminates the software connection to the Agilent E5270B and deallocates system resources. This function must be executed to close the instrument handle when the program is done using the Agilent E5270B.

Synopsis

```
int close_E5270(void)
```

Example

```
int err;
err = close_E5270();
```

See Also

- “open_E5270”

connect_pin

This function enables the channel specified by *port*. Also, this command assigns a *pin* number to the specified *port*. After this command, you can use *pin* number instead of *port* to specify the channel.

The connect_pin(0,0) function clears all assignments.

Synopsis

```
int connect_pin(port,pin)

int port, pin;
```

Arguments

Item	Description
port	Specify channel to enable. Available *port* values are 0, GNDU, SMU1 to SMU8, or values shown in Table 2-2. *port*=0 clears the port assignment for the specified *pin*.
pin	This item is necessary to keep compatibility with the Agilent 4070's connect_pin function. For the Agilent 4070, this item specifies switching matrix pin number to connect to the specified *port*. For the Agilent E5270B, the *pin* value can be used in other functions instead of the *port* number to specify the channel. Available *pin* values are 0 to 49. *pin* =0 clears the pin assignment for the specified *port*.

One *port* can be specified by multiple *pins* by executing multiple connect_pin functions. However, one *pin* can specify only one *port*. The most recent function is effective for the *pin*.

For the Agilent 4070, *pin*=1 to 48 specify the switching matrix pin numbers. *pin*=49 specifies the connector for chuck connection.

Example

```
int err;
err = connect_pin(SMU1,12);
```

See Also

- "disconnect_all"

disable_port

This function sets the specified channel to the zero output state as shown in Table 2-6.

Synopsis

```
int disable_port(port)

int port;
```

Arguments

Item	Description
port	Specify channel to be set to the zero output state by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 0 to 49, SMU1 to SMU8, or values shown in Table 2-2 on page 2-8. *port*=0 sets all ports to the zero output state.

Example

```
int err;
err = disable_port(0);
```

See Also

- "connect_pin"

Table 2-6 **Zero Output State**

Setup Item	Setting Value
Output Mode	Voltage output
Output Range	No change or 20 V if SMU was in I source mode.
Output Voltage	0 V
Current Compliance	100 mA

disconnect_all

This function disables all channels, and clears all assignments set by the connect_pin functions.

This function is equivalent to the connect_pin(0,0) function, and is recommended for better program readability.

Synopsis

```
int disconnect_all()
```

Example

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

See Also

- “connect_pin”

error_info

This function returns a TIS error number and an error message string.

This function checks the completion status of the most recently executed TIS function, and returns the error numbers and error message string to *errn* and *errm*, respectively.

Synopsis

```
int err_info(errn, errm)

int errn[2];
char errm[1024];
```

Arguments

Item	Description
errn	Pointer to an integer array in which to return the TIS error number. **errn[0]:** TIS error number **errn[1]:** CMS, DVM, or PGU error number that is meaningful for the Agilent 4070, but meaningless for the Agilent E5270B. Always 0 is returned. Must be a pointer to an integer array with size = 2.
errm	Pointer to a character array in which to return the TIS error message. A NULL terminated string is returned to the array. Must be a pointer to an character array with size 1024. Or you can specify NULL for *errm* if you want to discard the message.

The first element of the *errn* array is the TIS error number. The second element is necessary to keep compatibility with the Agilent 4070's error_info function.

If the init_system function is executed, both error numbers are set to 0, and the error message string is set to a null string.

No value is returned to the function itself, even if an error occurs.

Example

```
int err, errn[2];
char errm[1024];
err = error_info(errn, errm);
```

force_i

This function forces the specified current from the specified SMU.

Synopsis

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

int port;
double current, range, compliance;
```

Arguments

Item	Description
port	Specify current output channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
current	Value of current to force. Numeric expression [A].
range	Output ranging mode. Numeric expression [A]. See Table 2-3 on page 2-9 to know the available values.
compliance	Voltage compliance value. Numeric expression [V]. The *compliance* sets the maximum output voltage of an SMU operating as a current source (I_SOURCE mode). A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the last output mode of this channel was I_SOURCE, or 20 V if the last output mode was V_SOURCE.

NOTE current and compliance values

See Table 2-5 on page 2-12 to know the available *current* values, its setting resolution, and the maximum *compliance* value for the range actually used.

Example

```
int err;
double current, range, compliance;

   : : : : :

err = force_i(SMU1, current, range, compliance);
```

force_v

This function forces the specified voltage from the specified SMU.

Synopsis

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

int port;
double voltage, range, compliance;
```

Arguments

Item	Description
port	Specify voltage output channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
voltage	Value of voltage to force. Numeric expression [V].
range	Output ranging mode. Numeric expression [V]. See Table 2-3 on page 2-9 to know the available values.
compliance	Current compliance value. Numeric expression [A]. The *compliance* sets the maximum output current of an SMU operating as a voltage source (V_SOURCE mode). A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the last output mode of this channel was V_SOURCE, or 100 µA if the last output mode was I_SOURCE.

NOTE voltage and compliance values

See Table 2-4 on page 2-11 to know the available *voltage* values, its setting resolution, and the maximum *compliance* value for the range actually used.

Example

```
if (force_v(SMU1, 10.0, 20.0, .05) == -1) error_rep();
```

init_system

This function initializes the Agilent E5270B.

Execute the init_system function before executing any other TIS function.

Synopsis

```
int init_system()
```

Example

```
if (init_system() == -1) error_rep();
```

measure_bdv

This function triggers quasi-pulsed measurements to measure breakdown voltage, then returns the breakdown voltage. If this function finishes successfully, a 0 is returned. If not, -1 is returned.

The conditions of the quasi-pulsed measurements are determined by the set_bdv function. See “set_bdv” on page 2-39.

Synopsis

```
int measure_bdv(voltage, status, interval)

double *voltage;
int *status,interval;
```

Arguments

Item	Description
voltage	Specify a pointer to double variable in which to return the measured voltage. The measured value (breakdown voltage) is returned.
status	Specify a pointer to integer variable in which to return the measurement status code. Returns one of the status code shown in Table 2-7.
interval	Specify measurement interval, BDV_INTVL_SHORT (0) or BDV_INTVL_LONG (1).

NOTE

Interval

The *interval* defines how often the voltage gradient is monitored (calculated). If the *interval* is set to BDV_INTVL_SHORT, the gradient is calculated after each voltage measurement. If the *interval* is set to BDV_INTVL_LONG, the gradient is calculated after every 10 voltage measurements.

When the *interval* is set to BDV_INTVL_SHORT, a slew rate of less than 1 V/ms causes an error (*status* = BDV_TOO_SLOW).

When the *interval* is set to BDV_INTVL_LONG, a slew rate of less than 0.1 V/ms causes an error (*status* = BDV_TOO_SLOW).

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

Table 2-7 **Status Code of measure_bdv**

Status Code [a]		Description
0	BDV_NORMAL	The quasi-pulsed measurement ended normally.
1	BDV_COMP_OTHER	Another unit reached compliance.
2	BDV_CURRENT_LOW	Breakdown voltage measurement failed because current did not reach breakdown *current* within the stop voltage specified by set_bdv function.
3	BDV_OSC	This unit is oscillating.
4	BDV_OVERFLOW	Measurement overflow occurred while monitoring output voltage or measuring breakdown voltage.
6	BDV_TIMEOUT	The quasi-pulsed measurement did not reach breakdown *current* within time-out.
7	BDV_TOO_SLOW	The monitored slew rate of the output voltage is too small.

a. One of the above integers is returned to *status*. You can also use the associated macros in your program to determine which status was returned.

Example

```
int err, st;
double range, start, stop, current, hold, delay, v_meas;

   : : : : :

err = set_bdv(SMU1, range, start, stop, current, hold, delay);

if (measure_bdv(&v_meas, &st, BDV_INTVL_SHORT)==-1) error_rep();
if(st > 5)
if (measure_bdv(&v_meas, &st, BDV_INTVL_LONG)==-1) error_rep();
```

See Also

- “set_bdv”

measure_i, measure_it

These functions measure DC current using a specified SMU and returns the measurement value in amperes.

Synopsis

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

int measure_it(port, current, range, time_stamp)
int port;
double *current, range, *time_stamp;
```

Arguments

Item	Description
port	Specify current measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
current	Specify a pointer to double variable in which to return the measurement value.
range	Measurement ranging mode. Numeric expression [A]. See Table 2-3 on page 2-9 to know the available values. If the SMU is I_SOURCE mode, set any value. [a]
time_stamp [b]	Specify a pointer to double variable in which to return the time stamp value. Returns accumulated time (in seconds) from when the timer count is cleared until the measurement is completed.

a. This is a dummy parameter. Measurement range is actually set to same as output current range. Refer to force_i function for the output range.
b. Only for the measure_it function. The data will be returned if the time stamp data output is enabled by the set_timestamp function.

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

Example

```
double current;
if (measure_i(SMU1, &current, 1E-3) == -1) error_rep();
```

See Also

- “reset_timestamp”

measure_ileak

This function triggers the quasi-pulsed measurement to measure leakage current according to the conditions set by the set_ileak function, then returns the measurement value to the *current* variable. See “set_ileak” on page 2-41.

Synopsis

```
int measure_ileak(port, current, status, interval)

int port, *status, interval;
double *current;
```

Arguments

Item	Description
port	Specify current measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
current	Specify a pointer to double variable in which to return the measured current. The measured value (leakage current) is returned.
status	Specify a pointer to integer variable in which to return the measurement status code. Returns one of the status code shown in Table 2-8.
interval	Specify measurement interval, ILEAK_INTVL_SHORT (= 0) or ILEAK_INTVL_LONG (= 1).

NOTE

Interval

The *interval* defines how often the voltage gradient is monitored (calculated). If the *interval* is set to ILEAK_INTVL_SHORT, the gradient is calculated after each voltage measurement. If the *interval* is set to ILEAK_INTVL_LONG, the gradient is calculated after every 10 voltage measurements.

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* = ILEAK_TOO_SLOW). 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* = ILEAK_TOO_SLOW).

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

Table 2-8 **Status Code of measure_ileak**

Status Code [a]		Condition
0	ILEAK_NORMAL	The quasi-pulsed measurement ended normally.
1	ILEAK_COMP_OTHER	Another unit reached compliance.
2	ILEAK_COMP	This unit reached compliance.
3	ILEAK_OSC	This unit is oscillating.
4	ILEAK_OVERFLOW	Measurement overflow occurred while monitoring the output voltage or measuring the leakage current.
6	ILEAK_TIMEOUT	The quasi-pulsed measurement did not reach *output* (specified by set_ileak) within the time-out.
7	ILEAK_TOO_SLOW	The monitored slew rate of the output voltage is too small.

a. One of the above integers is returned to *status*. You can also use the associated macros in your program to determine which status was returned.

Example

```
int err, st;
double range, voltage, comp, start, hold, delay, ileak;

   : : : : :

err = set_ileak(SMU1, range, voltage, comp, start, hold, delay);

if (measure_ileak(SMU2, &ileak, &st, ILEAK_INTVL_SHORT)==-1)
error_rep();
if (st>5)
if (measure_ileak(SMU2, &ileak, &st, ILEAK_INTVL_LONG)==-1)
error_rep();
```

See Also

- “set_ileak”

measure_m

This function performs multi channel DC voltage or current measurements using up to eight SMU channels. The channels perform measurement sequentially in the order defined in the *ports*.

Before this function, execute force_i or force_v function to set the measurement mode for each channel. The force_i function sets the channel to voltage measurement mode, and force_v sets the channel to current measurement mode.

Synopsis

```
int measure_m(n, ports, meas_vals, ranges, time_stamps)

int n, ports[n];
double meas_vals[n], ranges[n], time_stamps[n];
```

Arguments

Item	Description
n	Specify how many ports will perform measurement. The size of the *ports*, *meas_vals*, *ranges*, and *time_stamps* arrays must be *n*. *n* can be from 1 to 8.
ports	Specify a pointer to an integer array that contains the measurement channels. This array must have *n* elements. The array should contain the port addresses or pin numbers that are assigned to the port by the connect_pin function. Available values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
meas_vals	Specify a pointer to a double array in which to return the measured values.This array must have *n* elements. Measurement results are returned to *meas_vals* array in order that corresponds to elements in the *ports* array.

Item	Description
ranges	Specify a pointer to a double array that contains the measurement ranging mode for each channel. This array must have *n* elements. Specify values in *ranges* array in order that corresponds to elements of the *ports* array. See Table 2-3 on page 2-9 to know the available values.
time_stamps [a]	Specify a pointer to a double array in which to return the time stamps for measured values. Returns accumulated time (in seconds) from when the timer count is cleared until the measurement is completed.

a. The data will be returned if the time stamp data output is enabled by the set_timestamp function.

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

Example

```
int err, n=3, ports[3];
double meas_vals[3], ranges[3], time_stamps[3];

ports[0]=8; ports[1]=12; ports[2]=16;
ranges[0]=0.0; ranges[1]=0.0; ranges[2]=0.0;

   : : : : :

err = measure_m(n, ports, &meas_vals, ranges, &time_stamps);
```

See Also

- “measure_i, measure_it”, “measure_v, measure_vt”, and “reset_timestamp”

measure_p

This function performs a pulsed spot measurement according to conditions set by the set_pbias function, and returns the measurement value.

Synopsis

```
int measure_p(port, mode, value, range)

int port, mode;
double *value,range;
```

Arguments

Item	Description
port	Specify measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8. If you specify 0, no measurement is executed and this function just triggers to force the pulsed bias that is set by set_pbias statement.
mode	Specify whether to perform a voltage or current measurement. Use one of the following macros (or values): **V_MEAS (1)** Voltage measurement mode. The channel must be a current source. **I_MEAS (2)** Current measurement mode. The channel must be a voltage source.
value	Specify a pointer to a double variable in which to return the measurement result.
range	Specify the measurement ranging mode. See Table 2-3 on page 2-9 to know the available values. In this function, 0 does not set the auto ranging mode. 0 sets the measurement range to the current compliance setting.

NOTE

A/D Converter

The pulsed spot measurement always uses high-speed ADC. And the number of averaging samples is set to 1. The resolution of returned value is (measurement range value)/20000.

Example

```
int err;
double current;

   : : : : :

err = set_pbias(SMU1, V_SOURCE, 20, -3, 15, 0.4, 1, 0.5, 1E-3);
err = measure_p(SMU1, I_MEAS, &current, 1E-3);
```

See Also

- “set_pbias”

measure_v, measure_vt

The function measures DC voltage using the specified SMU and returns the measurement value to the *voltage* variable.

Synopsis

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

int measure_vt(port, voltage, range, time_stamp)
int port;
double *voltage, range, *time_stamp;
```

Arguments

Item	Description
port	Specify measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
voltage	Specify a pointer to a double variable in which to return the measurement result.
range	Measurement ranging mode. Numeric expression [V]. See Table 2-3 on page 2-9 to know the available values. If the SMU is V_SOURCE mode, set any value. [a]
time_stamp [b]	Specify a pointer to double variable in which to return the time stamp value. Returns accumulated time (in seconds) from when the timer count is cleared until the measurement is completed.

a. This is a dummy parameter. Measurement range is actually set to same as output voltage range. Refer to force_v function for the output range.
b. Only for the measure_vt function. The data will be returned if the time stamp data output is enabled by the set_timestamp function.

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

Example

```
double voltage1;
if (measure_v(SMU2, &voltage1, 1.8) == -1) error_rep();
```

See Also

- “reset_timestamp”

port_status, port_status_t

This function returns the status of the specified port. This status indicates the condition of port after most recent measurement was performed by port.

Synopsis

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

int port_status_t(port, status, time_stamp)
int port, *status;
double *time_stamp;
```

Arguments

Item	Description
port	Specify measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, GNDU, and values shown in Table 2-2 on page 2-8.
status	Specify a pointer to integer variable in which to return the status code. Returns one of the status code shown in Table 2-9.
time_stamp [a]	Specify a pointer to double variable in which to return the time stamp value. Returns accumulated time (in seconds) from when the timer count is cleared until the last measurement is completed. If the port is not a measurement port, such as GNDU, 0.0 is always returned to *time_stamp*.

a. Only for the port_status_t function. The data will be returned if the time stamp data output is enabled by the set_timestamp function.

NOTE

status

If two or more conditions occur during a measurement, the priority of the returned numerical code is the following order:

`DCS_OSC > DCS_OVERFLOW > DCS_COMP > DCS_COMP_OTHER`

For example, if the SMU reached compliance and was oscillating during the measurement, `DCS_OSC` (oscillating) is returned to the *status* variable. This *status* variable remains unchanged until the next measurement.

Table 2-9 **Status Code of port_status and port_status_t**

Status Code [a]		Description
0	NORMAL_MEAS [b]	Normal.
1	DCS_COMP_OTHER	Another unit has reached compliance.
2	DCS_COMP	This unit has reached compliance.
3	DCS_OSC	This unit is oscillating.
4	DCS_OVERFLOW	Measurement overflow.
5	DCS_SWP_STOPPED	Sweep was aborted by compliance condition. (stop mode = 2 in set_iv or set_piv)

a. One of the above integers is returned to *status*. You can also use the associated macros in your program to determine which status was returned.
b. The returned status of a port that has not performed measurement is always NORMAL_MEAS (normal).

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

Example

```
int smu_status;
if (port_status(SMU1,&smu_status)== -1) error_rep();
```

open_E5270

This function establishes the software connection to the Agilent E5270B. This function must be executed before using the E5270B TIS functions.

Synopsis

```
int open_E5270(*instr_desc, err_detect, *log_file)

char *instr_desc, *log_file;
int err_detect;
```

Arguments

Item	Description
instr_desc	Specify the GPIB interface and address of the Agilent E5270B. For example, *instr_desc*="GPIB0::17::INSTR" specify the Agilent E5270B that is the address 17 on GPIB0 interface.
err_detect	Specify automatic instrument error checking ON or OFF by using the following macros (or values): **ERR_DETECT_OFF (0)** Disables error checking. **ERR_DETECT_ON (1)** Enables error checking. If error checking is enabled, the VXI*plug&play* driver will query the Agilent E5270B for an error at the end of each function call. Even if this function is disabled, the error checking for TIS library and VXI*plug&play* driver will be always performed.
log_file	Specify a log file name. If you want to record error log to a file, set file name to *log_file* (ex: "test1.log"). If you want to display error message on the computer screen, set *log_file*=NULL.

Example

```
open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON, "LOG1");
```

See Also

- "close_E5270"

NOTE Multiple E5270Bs cannot be opened at the same time in a program. Terminate the session by using the close_E5270 function before opening another one.

reset_timestamp

This function enables to use the force_i or force_v function to clear the timer count (time stamp). The timer count will be cleared when the force_i or force_v function is first executed after the reset_timestamp function.

To enable the time data output, execute the set_timestamp function.

Effects of the reset_timestamp and set_timestamp function will be disabled by the init_system function.

Synopsis

```
int reset_timestamp(void)
```

Example

```
int err;
err = reset_timestamp();
```

See Also

- “set_timestamp”

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

set_adc

The function sets the operation parameters of analog-to-digital converter (high speed ADC or high resolution ADC) that SMUs use for measurements.

To select which converter to use for an SMU, use set_smu_ch function.

Synopsis

```
int set_adc(adc, mode, value, autozero)

int adc, mode, autozero;
double value;
```

Arguments

Item	Description
adc	Specify the high speed ADC or high resolution ADC. You can use the following macros (or values). PERCH_ADC (0): High-speed ADC REF_ADC (1): High-resolution ADC.
mode	Specify the integration mode. See Table 2-10.
value	Specify integration time or number of averaging samples. See Table 2-10.
autozero [a]	Set auto-zero function ON or OFF. This function is available only for the high resolution ADC. You can use the following macros (or values). AUTOZERO_OFF (0): OFF AUTOZERO_ON (1): ON

a. If *adc* is PERCH_ADC, this is dummy parameter.

Example

```
int err;
err = set_smu_ch(0,PERCH_ADC,FILTER_OFF);
err = set_adc(PERCH_ADC,INTEG_MANUAL,10,0);
```

See Also

- “set_smu_ch”

Table 2-10 mode and value

adc	mode		value
High speed ADC	0	INTEG_MANUAL	Specify the number of averaging samples. 1 to 1023. *value*=0 sets the number of samples to 1.
	1	INTEG_SHORT	Specify *value* in the following formula: [a] *Number of samples=Initial averaging* × *value* Available values are 1 to 1023. *value*=0 sets the number of samples to *Initial averaging*.
	2	INTEG_MEDIUM	Number of averaging samples is always 128. If you set *value*, the value is ignored.
	3	INTEG_LONG	Specify *value* in the following formula: *Number of samples*=128 × *value* Available values are 1 to 100. *value*=0 sets the number of samples to 128 × 16.
High resolution ADC	0	INTEG_MANUAL	Specify the integration time. 80E-6 to 10.16E-3 sec. Or specify the number of power line cycles for integration by ADC. 1 to 100. *value*=0 sets the integration time to 240E-6 sec.
	1	INTEG_SHORT	Specify the integration time. 80E-6 to 10.16E-3 sec. *value*=0 sets the integration time to 480E-6 sec.
	2	INTEG_MEDIUM	Integration time is always 20 msec for 50 Hz line frequency site, or 16.66 msec for 60 Hz site. If you set *value*, the value is ignored.
	3	INTEG_LONG	Specify the number of power line cycles for integration by ADC. 1 to 100. *value*=0 sets the number of power line cycles to 16.

a. In the formula, *Initial averaging* indicates the number of averaging samples automatically set by the Agilent E5270B and you cannot change.

set_bdv

This function sets quasi-pulsed spot measurement parameters for breakdown voltage measurement. To start this measurement, use the measure_bdv function.

Synopsis

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

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

Arguments

Item	Description
port	Specify source/measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
range	Output ranging mode. Numeric expression [V]. See Table 2-3 on page 2-9 to know the available values.
start, *stop*	Specify the start voltage or stop voltage of the search. Numeric expression [V]. The difference between the *start* voltage and *stop* voltage must be 10 V or more.
current	Specify the breakdown current. Numeric expression [A]. The *current* specifies the current compliance of the SMU. When breakdown occurs, the current will increase rapidly and reach the current compliance (*current*) quickly. This limits the current to prevent damage to the DUT.
hold	Specify the hold time to keep *start* voltage. Numeric expression [s]. 0 to 655.35, 0.01 resolution.
delay	Specify the delay time to keep measurement voltage before measurement is started. Numeric expression [s]. 0 to 6.5535, 0.0001 resolution.

NOTE

start, stop, and current values

See Table 2-4 on page 2-11 to know the available *start/stop* values, the setting resolution, and the maximum compliance (*current*) value for the range actually used.

See Also

- "measure_bdv"

NOTE

Breakdown Voltage Measurement

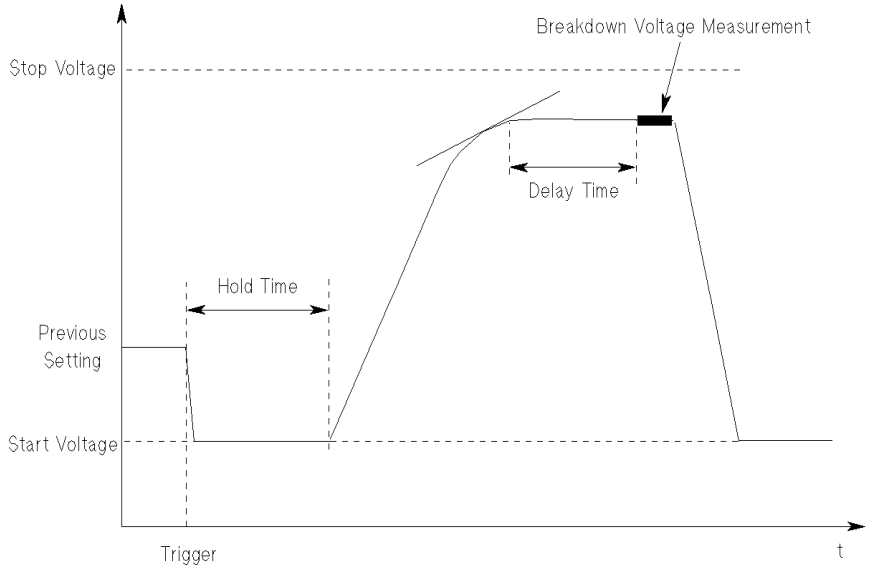
Figure 2-1 shows the quasi-pulsed wave form for a breakdown voltage measurement. After start trigger (measure_bdv function), the SMU forces *start* voltage, waits the *hold* time, and starts to change output voltage. The output value will be changed from *start* to *stop* in a constant rate. Where the rate of voltage change is determined by capacitance on the device under test (DUT), cables, test fixtures, and so on.

If breakdown occurs on the DUT when the SMU output voltage is being changed, the current will increase rapidly and reach the current compliance (*current*) quickly, and the rate of voltage change will be slower. And when the rate becomes less than half of the initial rate, the SMU waits the *delay* time and performs voltage measurement. After that, the SMU output voltage is set to *start*. The measurement result is returned to the *voltage* variable specified by the measure_bdv function.

If breakdown does not occur, the SMU stops voltage change at the *stop* value, and returns the output value to *start*.

This measurement function is effective for the DUT that is designed with a very wide breakdown voltage range. When breakdown occurs, the current increases, but is limited to your desired *current* setting. This will prevent the DUT from damage.

Figure 2-1 **Quasi-pulsed Wave form for Breakdown Measurement**



set_ileak

This function sets quasi-pulsed spot measurement conditions to measure leakage current. To start this measurement, use the measure_ileak function.

Synopsis

```
int set_ileak(port, range, voltage, compliance, start, hold,
delay)

int port;
double range, voltage, compliance, start, hold, delay;
```

Arguments

Item	Description
port	Specify source channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
range	Output ranging mode. Numeric expression [V]. See Table 2-3 on page 2-9 to know the available values.
voltage	Specify the voltage at which the leakage current is measured. Numeric expression [V]. The difference between *voltage* and *start* must be 10 V or more.
compliance	Specify the current compliance. Numeric expression [A]. A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the last output mode of this channel was VS, or 100 μA if the last output mode was IS.
start	Specify the ramp output start voltage. Numeric expression [V]. The difference between *start* and *voltage* must be 10 V or more. A macro REMAIN is also available. If you set REMAIN, the last valid voltage setting of the port is used.

NOTE voltage, compliance, and start values

See Table 2-4 on page 2-11 to know the available *voltage/start* values, the setting resolution, and the maximum *compliance* value for the range actually used.

Item	Description
hold	Specify the hold time to keep *start* voltage. Numeric expression [s]. 0 to 655.35, 0.01 resolution.
delay	Specify the delay time to keep measurement voltage before measurement is started. Numeric expression [s]. 0 to 6.5535, 0.0001 resolution.

NOTE

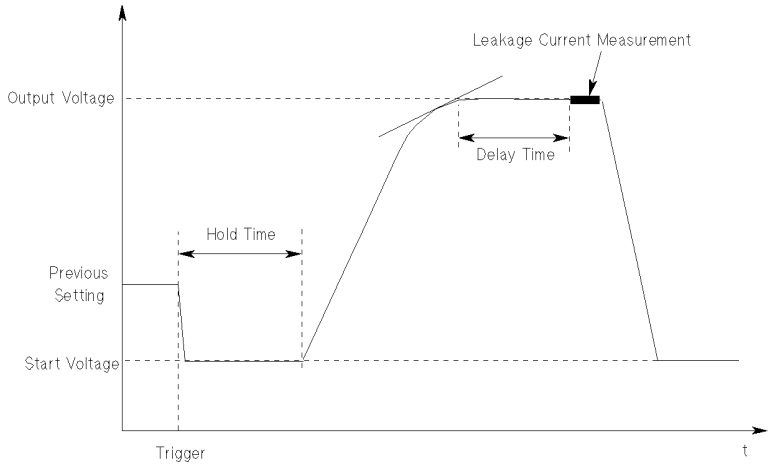
Leakage Current Measurement

Figure 2-2 shows the quasi-pulsed wave form for a leakage current measurement. After start trigger (measure_ileak function), the SMU forces the *start* voltage, waits *hold* time, and starts to change output voltage. The output value will be changed from *start* to *voltage* in a constant rate. Where the rate of voltage change is determined by capacitance on the device under test (DUT), cables, test fixtures, and so on.

The SMU monitors the output voltage. And when the output voltage comes near *voltage*, the rate of voltage change will be slower. And when the rate becomes less than half of the initial rate, the SMU waits the *delay* time and performs current measurement. After that, the SMU output voltage is set to *start*. The measurement result is returned to the *current* variable specified by the measure_ileak function.

This function is effective for a low leakage measurement with high voltage. The measurement voltage (*voltage*) is automatically detected, and measurement is performed soon, so high voltage will not be applied to the DUT for a long time. This will prevent the DUT from damage.

Figure 2-2 **Quasi-pulsed Wave form for Leakage Current Measurement**



See Also

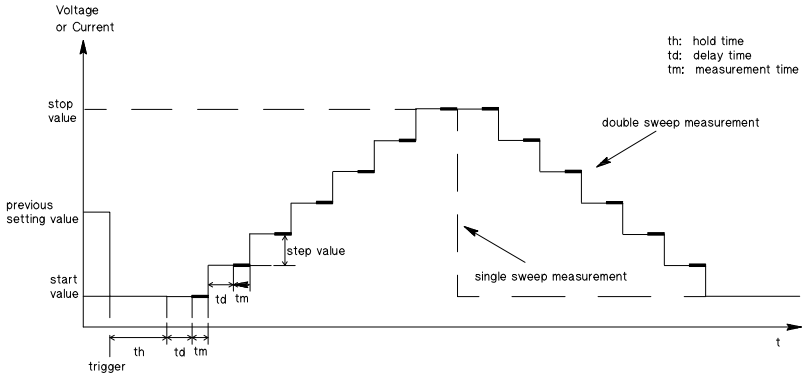
- "measure_ileak"

set_iv

This function sets the setup parameters for staircase sweep measurements that are started by the sweep_iv or sweep_miv function.

The setup parameters *start* (voltage or current), *stop*, *hold*, and *delay* determine the sweep measurement conditions as shown in Figure 2-3.

Figure 2-3 Staircase Sweep Setup Parameters



The start, stop, and step values in this figure are determined as shown in Table 2-11 and Table 2-12.

Table 2-11 start, stop, step values for linear sweep

Item	Description
start value	resolution: depends on the output range setting.
stop value [a]	stop = start + step × (number of steps − 1)
step value	step = (stop − start) / (number of steps − 1) resolution: depends on the output range setting. maximum step value: stop – start

a. The actual stop value may be slightly different from the specified value. Because the step value may be rounded off, depending on the setting resolution. See Table 2-4 on page 2-11 and Table 2-5 on page 2-12 for setting resolution.

Table 2-12 **start, stop, step values for log sweep**

Item	Description
start value	resolution: depends on the output range setting.
stop value [a]	stop = start × $1E^{N/20}$ where, N = step × (number of steps − 1)
step ratio	step ratio = A / (number of steps − 1) where, A = 20 × log (stop/start) resolution: 0.02 dB/decade maximum: 20 dB/decade

a. The actual stop value may be slightly different than the specified value. Because the step value may be rounded off, depending on the setting resolution. See Table 2-4 on page 2-11 and Table 2-5 on page 2-12 for setting resolution.

NOTE

Functions that disable the set_iv settings

If the set_piv function is executed after this function, the set_iv settings will not be effective, and the set_piv settings will be effective.

Also the set_iv settings will be cleared by the init_system function that initializes the Agilent E5270B.

Synopsis

```
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

NOTE

start, stop, and compliance values

See Table 2-4 on page 2-11 or Table 2-5 on page 2-12 to know the available *start/stop* values, the setting resolution, and the maximum *compliance* value for the range actually used.

Item	Description
port	Specify sweep source channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
sweep_mode	Specify sweep mode by using the following macros (or values): **LINEAR_V (1)** Linear voltage single sweep **LINEAR_I (2)** Linear current single sweep **LINEAR_V_DBL (3)** Linear voltage double sweep **LINEAR_I_DBL (4)** Linear current double sweep **LOG_V (−1)** Logarithmic voltage single sweep **LOG_I (−2)** Logarithmic current single sweep **LOG_V_DBL (−3)** Logarithmic voltage double sweep **LOG_I_DBL (−4)** Logarithmic current double sweep
range	Output ranging mode. Numeric expression. See Table 2-3 on page 2-9 to know the available values. Also see NOTE below.
start, *stop*	Sweep start and stop values. Numeric expression. For a log sweep, *start* and *stop* must be the same polarity, and cannot be 0.
number	Number of sweep steps from *start* to *stop*. 2 to 1001. For double sweep, *number* is effective for both sweep from *start* to *stop* and sweep from *stop* to *start*.

Item	Description
hold	Specify the hold time to keep the *start* value after the trigger. Numeric expression [s]. If a non-zero *delay* is set, an additional *delay* time follows after the *hold* time before starting measurement. 0 to 655.35, 0.01 resolution.
delay	Specify the delay time to keep each step value before starting measurement. Numeric expression [s]. 0 to 65.535, 0.0001 resolution.
compliance	Specify a current or voltage compliance value depending on the *sweep_mode*. Numeric expression. A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the output mode of this channel was same as the last output mode. Or, the compliance value is set to 100 μA if the last output mode was IS for the voltage sweep channel, or 20 V if the last output mode was VS for the current sweep channel.
power_compliance	Specify power compliance that limits the power (voltage × current being forced and measured) applied to the *port*. Numeric expression [W]. 0.001 to 4 [W], in 0.001 [W] resolution for MPSMU 0.001 to 20 [W], in 0.001 [W] resolution for HPSMU To disable the power compliance, set *power_compliance*=0.
stop_mode	Specify automatic sweep abort function ON or OFF by using the following macros (or values): **COMP_CONT (1)** Disables the function. **COMP_STOP (2)** Enables the function. If you set *power_compliance* to non-zero value, *stop_mode* setting is ignored. However, you cannot omit this parameter. Set *stop_mode* to COMP_CONT (1) or COMP_STOP (2). For the automatic sweep abort function, see the following NOTE.

NOTE

Source Output Range Operation

If you set *range*=0 for the linear sweep, the source channel uses the lowest output range that covers both *start* and *stop* values. The output range is not changed when the sweep measurement is being executed.

If you set *range*=0 for the log sweep, the source channel uses the optimum output range for the sweep step output value.

If you set non-zero *range* value for the voltage sweep, output range is fixed to the specified *range*.

If you set non-zero *range* value for the current sweep, and if the specified *range* covers both *start* and *stop* value, the specified *range* is used, else, the SMU automatically ranges up to range that can force both *start* and *stop* value. The output range is not changed when the sweep measurement is being executed.

NOTE

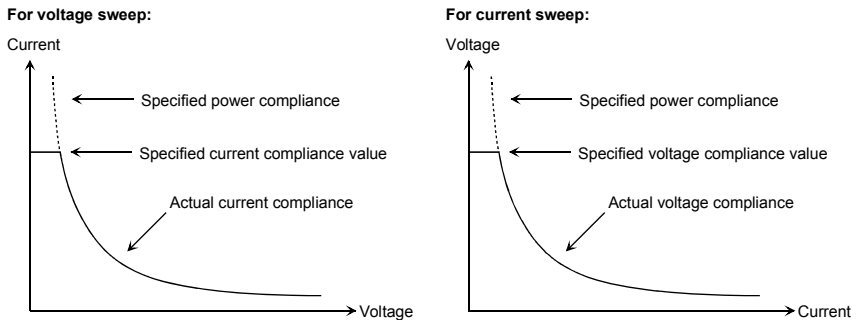
power_compliance and compliance

If you specify *power_compliance*=0, available voltage *compliance* or current *compliance* value depends on the maximum sweep output value (*start* or *stop*). See Table 2-4 on page 2-11 and Table 2-5 on page 2-12 for maximum *compliance* value.

If you specify non-zero *power_compliance* value, the SMU changes the current or voltage compliance value every sweep step. The value is the lower value of either the specified compliance value or the compliance value given by the following formula. See the following figure.

Current compliance = Specified *power_compliance* value / Step voltage

Voltage compliance = Specified *power_compliance* value / Step current



NOTE

Automatic Sweep Abort Function

If you set *stop_mode* to COMP_CONT, voltage or current sweep continues to the last specified step, even if one of the following sweep abort conditions occurs.

- The output reaches voltage compliance or current compliance
- A measurement value exceeds the specified measurement range
- An SMU oscillates

If you set *stop_mode* to COMP_STOP, the sweep stops when one of the sweep abort conditions occurs.

If you set *power_compliance* to non-zero value, *stop_mode* is ignored. The sweep stops when one of the sweep abort conditions occurs or the output reaches *power compliance*.

After sweep was aborted, the sweep_iv, sweep_miv, or status_miv function returns data shown in the following example that an abort condition occurs at Step=N.

Step	measure or meas_vals	source	sync	statuses	times
1	(measured data)	(output data)	(output data)	0	(time data)
:	:	:	:	:	:
N-1	(measured data)	(output data)	(output data)	0	(time data)
N	(measured data)	(output data)	(output data)	X[a]	(time data)
N+1	9999999.99999	0	0	5	0
:	:	:	:	:	:
Last	9999999.99999	0	0	5	0

[a] X will be a status code that indicates a sweep abort condition.

Example

```
int nop;
double range, start, stop, hold, delay, comp, pcomp, meas, swp1;

   : : : : :

set_iv(SMU1, LINEAR_V, range, start, stop, nop, hold, delay, comp,
pcomp, COMP_STOP);

sweep_iv(SMU1, I_MEAS, 0.0, meas, swp1, NULL);
```

See Also

- “sweep_iv”
- “sweep_miv”
- “set_sync”
- “set_pbias”

set_pbias

This function sets the pulse source parameters for a pulsed spot measurement that is started by the measure_p function or a staircase sweep with pulsed bias measurement that is started by the sweep_iv function.

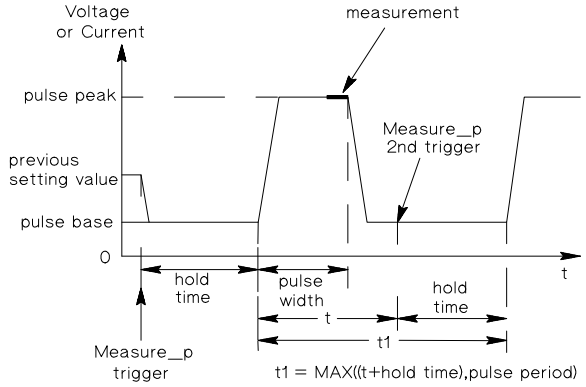
The setup parameters *base*, *peak*, *width*, *period*, and *hold* determine the pulsed spot measurement conditions as shown in Figure 2-4 and the staircase sweep with pulsed bias measurement as shown in Figure 2-5.

To start pulsed spot measurements, execute this function and then measure_p function. The SMU forces the *base* value, waits the *hold* time, and forces a pulse. During the *period*, the SMU cannot force next pulse. The SMU can force next pulse in the period given by the following *t1* value.

- *t1* = MAX(*t*+*hold*, *period*)

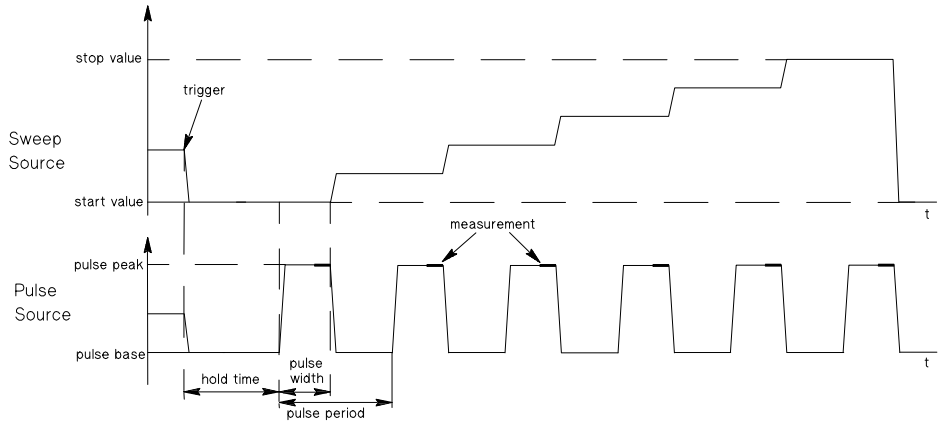
where *t* is time from beginning of pulse to next measurement trigger.

Figure 2-4 Pulsed Spot Measurements



The measurement channel performs measurement so that the pulse width and pulse period are kept (the integration time setting is ignored). Only one channel can be used for measurement.

Figure 2-5 **Staircase Sweep with Pulsed Bias Measurements**



To start staircase sweep with pulsed bias measurement, execute the set_iv function, set_pbias function (this function), and then sweep_iv function. The sweep source ignores the hold time and delay time defined by the set_iv function, and changes the sweep output value in the timing set by this function. See Figure 2-5.

NOTE

Functions that disable the set_pbias settings

If the set_iv function is executed after this function, the set_pbias settings will be cleared, and the set_iv settings will be effective.

If program contains this function and the set_piv function, the set_pbias settings will not be effective, and the set_piv settings will be effective.

If the disable_port function is executed for the same *port* set to this function, the set_pbias settings will be cleared.

Also the set_pbias settings will be cleared by the init_system function that initializes the Agilent E5270B.

Synopsis

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

int port, mode;
double base, peak, width, period, hold, compliance;
```

Arguments

NOTE base, peak, and compliance values

See Table 2-4 on page 2-11 or Table 2-5 on page 2-12 to know the available *base/peak* values, the setting resolution, and the maximum *compliance* value for the range actually used.

Item	Description
port	Specify pulse source channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
mode	Specify output mode by using the following macros (or values): **V_SOURCE (1)** Voltage output mode **I_SOURCE (2)** Current output mode
range	Output ranging mode. Numeric expression. See Table 2-3 on page 2-9 to know the available values. Also see NOTE below.
base, *peak*	Specify the pulse base and peak values of the voltage or current being forced. If *mode* is I_SOURCE, *base* and *peak* must have the same polarity.
width	Pulse width. Numeric expression. 0.0005 to 2 [s] in 1E−4 [s] resolution.
period	Pulse period. Numeric expression. 0.005 to 5 [s] in 1E-4 [s] resolution. Restrictions: • *period* ≥ *width* +2 msec (for *width* ≤ 100 ms) • *period* ≥ *width* +10 msec (for *width* > 100 ms) If you set *period*=0, the E5270B automatically sets the pulse period to 5 msec (for *width* ≤ 3 ms), *width* +2 msec (for 3 ms < *width* ≤ 100 ms), or *width* +10 msec (for *width* > 100 ms). If you do not specify *period*, 0 sec is set.

Item	Description
hold	Specify time to wait after trigger before forcing *peak* value. Numeric expression. 0 to 655.35 [s] in 0.01 [s] resolution.
compliance	Specify a current or voltage compliance value depending on the *mode*. Numeric expression. A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the output mode of this channel was same as the last output mode. Or, the compliance value is set to 100 μA if the last output mode was IS for the voltage sweep channel, or 20 V if the last output mode was VS for the current sweep channel.

Example

```
int nop;
double vstart, vstop, hold, delay, comp, pcomp;
double base, peak, width, period, pulse_hold, pulse_comp;
double meas[101], swp1[101];

   : : : : :

set_iv(SMU1, LINEAR_V, 20.0, vstart, vstop, nop, hold, delay,
comp, pcomp, COMP_STOP);

set_pbias(SMU2, V_SOURCE, 20.0, base, peak, width, period,
pulse_hold, pulse_comp);

sweep_iv(SMU1, I_MEAS, 0.0, meas, swp1, NULL);
```

See Also

- “measure_p”
- “sweep_iv”

NOTE

Pulse Output Range Operation

If you set *range*=0, the pulse source channel forces the current or voltage pulse at the lowest range that includes both *base* and *peak* values.

If you set non-zero *range* value for the voltage pulse source, output range is fixed to specified range.

If you set non-zero *range* value for the current pulse source, output range is determined by limited auto-ranging mode. The SMU forces the current pulse at the specified range if the specified range can force both *base* and *peak* value. If not, the SMU ranges up to range that can force the *base* and *peak*.

set_piv

This function sets the setup parameters for pulsed sweep measurements that are started by the sweep_iv function.

The setup parameters pulse *base* (voltage or current), *start*, *stop*, *hold*, *width*, and *period* determine the pulsed sweep measurement conditions as shown in Figure 2-6 and Figure 2-7.

Figure 2-6 **Single Pulsed Sweep**

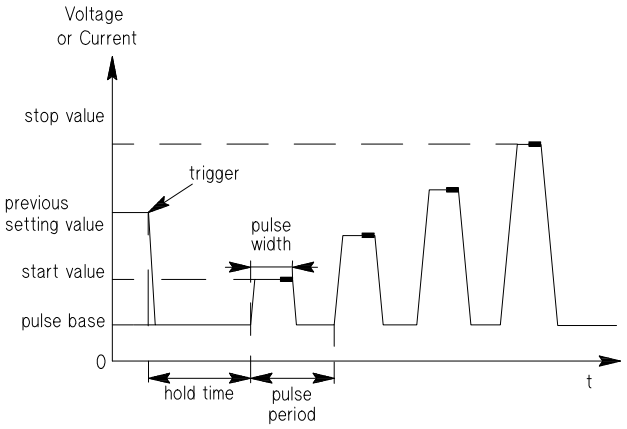
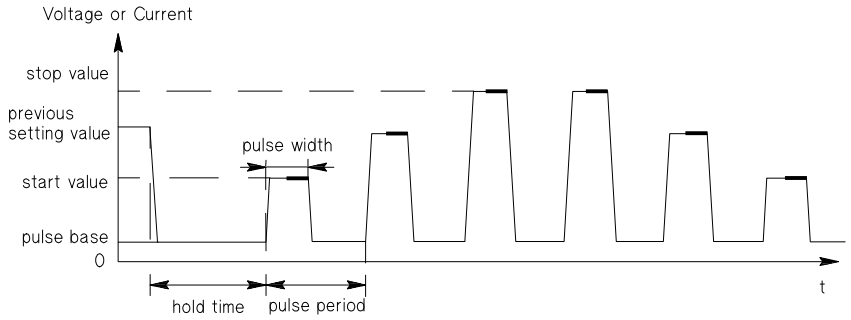


Figure 2-7 **Double Pulsed Sweep**



The start, stop, and step values are determined as shown in Table 2-13.

Table 2-13 **start, stop, step values for pulsed sweep measurement**

Item	Description
start value	resolution: depends on the set voltage or current range.
stop value [a]	stop value = start value + step value × (number of steps − 1)
step value	step value = (stop value − start value) / (number of steps − 1) resolution: depends on the set voltage or current range. maximum: *stop value − start value*

a. The actual stop value may be slightly different from the specified value. Because the step value may be rounded off, depending on the setting resolution. See Table 2-4 on page 2-11 and Table 2-5 on page 2-12 for setting resolution.

NOTE Functions that disable the set_piv settings

If the set_iv function is executed after this function, the set_piv settings will not be effective, and the set_iv settings will be effective.

Also the set_piv settings will be cleared by the init_system function that initializes the Agilent E5270B.

Synopsis

```
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

NOTE base, start, stop, and compliance values

See Table 2-4 on page 2-11 or Table 2-5 on page 2-12 to know the available *base/start*/stop values, the setting resolution, and the maximum compliance value for the range actually used.

Item	Description
port	Specify pulse sweep channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
sweep_mode	Specify sweep mode by using the following macros (or values): **LINEAR_V (1)** Linear voltage single sweep **LINEAR_I (2)** Linear current single sweep **LINEAR_V_DBL (3)** Linear voltage double sweep **LINEAR_I_DBL (4)** Linear current double sweep
range	Output ranging mode. Numeric expression. See Table 2-3 on page 2-9 to know the available values. Also see NOTE below.
base, *start*, *stop*	Specify the pulse base, start, and stop values of the voltage or current being forced. If *mode* is I_SOURCE, *base* and *start*/*stop* must have the same polarity.
number	Number of sweep steps from *start* to *stop*. 2 to 1001. For double sweep, *number* is effective for both sweep from *start* to *stop* and sweep from *stop* to *start*.
width	Pulse width. Numeric expression. 0.0005 to 2 [s] in 1E−4 [s] resolution.
period	Pulse period. Numeric expression. 0.005 to 5 [s] in 1E-4 [s] resolution. Restrictions: • *period* ≥ *width* +2 msec (for *width* ≤ 100 ms) • *period* ≥ *width* +10 msec (for *width* > 100 ms) If you set *period*=0, the E5270B automatically sets the pulse period to 5 msec (for *width* ≤ 3 ms), *width* +2 msec (for 3 ms < *width* ≤ 100 ms), or *width* +10 msec (for *width* > 100 ms). If you do not specify *period*, 0 sec is set.

Item	Description
hold	Specify time to wait after trigger before forcing *peak* value. Numeric expression. 0 to 655.35 [s] in 0.01 [s] resolution.
compliance	Specify a current or voltage compliance value depending on the *mode*. Numeric expression. A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the output mode of this channel was same as the last output mode. Or, the compliance value is set to 100 μA if the last output mode was IS for the voltage sweep channel, or 20 V if the last output mode was VS for the current sweep channel.
stop_mode	Specify automatic sweep abort function ON or OFF by using the following macros (or values): **COMP_CONT (1)** Disables the function. **COMP_STOP (2)** Enables the function. For the automatic sweep abort function, see the following NOTE.

NOTE

Pulse Sweep Output Range Operation

If you set *range*=0, the pulse sweep source forces the current or voltage pulse for each sweep step by using the lowest range that can force *base*, *start*, and *stop*.

If you set non-zero *range* value for the voltage pulse source, output range is fixed to specified range.

If you set non-zero *range* value for the current pulse source, output range is determined by limited auto-ranging mode. The SMU forces the current pulse at the specified range if the specified range can force *base*, *start*, and *stop*. If not, the SMU ranges up to range that can force *base*, *start*, and *stop*.

NOTE

Automatic Sweep Abort Function

If you set *stop_mode* to COMP_CONT, voltage or current sweep continues to the last specified step, even if one of the following sweep abort conditions occurs.

- The output reaches voltage compliance or current compliance
- A measurement value exceeds the specified measurement range
- An SMU oscillates

If you set *stop_mode* to COMP_STOP, the sweep stops when one of the sweep abort conditions occurs.

After sweep was aborted, the sweep_iv, sweep_miv, or status_miv function returns data shown in the following example that an abort condition occurs at Step=N.

Step	measure or meas_vals	source	sync	statuses	times
1	(measured data)	(output data)	(output data)	0	(time data)
:	:	:	:	:	:
N-1	(measured data)	(output data)	(output data)	0	(time data)
N	(measured data)	(output data)	(output data)	X[a]	(time data)
N+1	9999999.99999	0	0	5	0
:	:	:	:	:	:
Last	9999999.99999	0	0	5	0

[a] X will be a status code that indicates a sweep abort condition.

Example

```
int nop=11;
double vbase, vstart, vstop, width, period, hold, comp;
double meas[11], swp1[11];

   : : : : :

set_piv(SMU1, LINEAR_V, 20.0, vbase, vstart, vstop, nop, width,
period, hold, comp, COMP_STOP);

sweep_iv(SMU1, I_MEAS, 0.0, meas, swp1, NULL);
```

See Also

- “sweep_iv” and “set_sync”

set_smu_ch

This function selects which A/D converter the specified SMU uses to perform measurement, and whether the SMU filter is ON or OFF.

Synopsis

```
int set_smu_ch(port, adc, filter)
int port, adc, filter;
```

Argument

Item	Description
port	Specify SMU by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8. *port*=0 is also available. *port*=0 selects all SMUs.
adc	Specify A/D converter by using the following macros (or values). PERCH_ADC (0): High-speed ADC REF_ADC (1): High-resolution ADC.
filter	Specify filter setting by using the following macros (or values): **FILTER_OFF (0)** Disables filter for the specified SMU. **FILTER_ON (1)** Enables filter for the specified SMU.

NOTE

Filter

The initial filter setting is OFF, which enables higher speed measurement. Usually, this setting is OK because the SMU normally does not generate an overshoot voltage or current.

If the measurement result is not correct or DUT is damaged during measurement, set the *filter* to ON, which reduces overshoot voltage or current in the output of the SMU.

Example

```
int err;
err = set_smu_ch(0,PERCH_ADC,FILTER_OFF);
err = set_adc(PERCH_ADC,INTEG_MANUAL,10,0);
```

See Also

- “set_adc”

set_sync

The set_sync function can be used *after* set_iv or set_piv to set up a synchronous sweep source. The set_iv or set_piv function sets up the primary sweep source, and set_sync sets up the synchronous sweep source.

A synchronous sweep source can be set up for the following types of measurements:

- Staircase sweep measurements (set_iv)
- Staircase sweep with pulsed bias measurements (set_iv and set_pbias)
- Pulsed sweep measurements (set_piv)

If set_iv is used, the synchronous sweep measurement is started by sweep_iv or sweep_miv function. If set_piv is used, the synchronous sweep measurement is started by the sweep_iv function.

The output mode of the synchronous sweep source is automatically set to the same *sweep_mode* (linear or logarithmic, VS or IS, and double or single) as the primary sweep port set by the set_iv or set_piv function.

The synchronous sweep source always performs staircase sweep output even if the primary sweep source performs pulsed sweep output.

Note that you can specify only one port for the synchronous sweep source.

NOTE

Functions that clear the set_sync settings

The following functions clear the settings of this function. If one of the following functions is executed, the set_sync settings are lost. So, execute this function after these functions.

```
init_system
set_iv
set_piv
```

Synopsis

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

int port, mode;
double offset, ratio, compliance, power_compliance;
```

Arguments

Item	Description
port	Specify SMU for synchronous sweep source by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
mode	Specify the polarity of synchronization by using the following macros (or values): **POS_SYNC (0)** Positive synchronization. **NEG_SYNC (1)** Negative synchronization. The synchronous sweep source output is decided by *mode*, *offset*, *ratio*, and the primary sweep source output value as shown below: • *mode*=POS_SYNC: start value = *offset* + *ratio* × *start* of primary sweep source stop value = *offset* + *ratio* × *stop* of primary sweep source • *mode*=NEG_SYNC: start value = *offset* − *ratio* × *start* of primary sweep source stop value = *offset* − *ratio* × *stop* of primary sweep source where *offset* value is not effective for the logarithmic sweep. Specify *mode*, *offset*, and *ratio* values so that the synchronous sweep source start and stop values do not exceed the maximum output value of the SMU specified by *port*. Number of sweep steps for the synchronous sweep port is same as *number* that is set to the set_iv or set_piv function.
offset	Only for linear sweep. Specify offset used to calculate the synchronous sweep source output. Numeric expression.
ratio	Specify ratio used to calculate the synchronous sweep source output. Numeric expression. 0.01 to 10, 0.01 resolution. To use negative ratio, set *mode*=NEG_SYNC.

Item	Description
compliance	Specify a current or voltage compliance value for the synchronous sweep source. It depends on the *sweep_mode* of the primary sweep source. Numeric expression. A macro REMAIN is also available. If you set REMAIN, the compliance value is automatically set to the last compliance value if the output mode of this channel was same as the last output mode. Or, the compliance value is set to 100 μA if the last output mode was IS for the voltage sweep channel, or 20 V if the last output mode was VS for the current sweep channel.
power_compliance	Specify power compliance that limits the power (voltage × current being forced and measured) applied to the *port*. Numeric expression. 0.001 to 4 [W] in 0.001 [W] resolution. for MPSMU 0.001 to 20 [W] in 0.001 [W] resolution. for HPSMU To disable the power compliance, set *power_compliance*=0.

NOTE

offset and compliance values

See Table 2-4 on page 2-11 or Table 2-5 on page 2-12 to know the available output values, the setting resolution, and the maximum *compliance* value for the range actually used. The *offset* value must be calculated by using the primary sweep output value and the output value you desire.

Example

```
double offset, ratio, comp2, meas[11], swp1[11], swp2[11];

   : : : : :

set_sync(SMU2, POS_SYNC, offset, ratio, comp2, 0.0);
sweep_iv(SMU2, I_MEAS, 0.0, meas, swp1, swp2);
```

See Also

- “set_iv”
- “sweep_iv”
- “sweep_miv”

set_timestamp

This function enables or disables the time stamp data output for the following functions:

- measure_it
- measure_m
- measure_vt
- port_status_t
- status_miv

If the time stamp data output is enabled, these functions return time stamp data with other returned data such as measurement data and status data. If the time stamp data output is disabled, these functions return 0 as the time stamp data.

To get the time stamp data, the set_timestamp function must be executed before these functions. For sweep measurement, the set_timestamp function must be executed before sweep_iv or sweep_miv function.

Synopsis

```
int set_timestamp(on)
int on;
```

Arguments

Item	Description
on	0: Disables time stamp data output. Other than 0: Enables time stamp data output.

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

status_miv

This function returns the status of the data measured by sweep_iv or sweep_miv function.

Synopsis

```
int status_miv(n, ports, statuses, times)

int n, ports[n], statuses[nx];
double times[nx];
```

Where *x* is the *number* of steps specified in set_iv function. For double sweep, *x* should be 2*x*.

Arguments

Item	Description
n	Specify the number of measurement channels. 1 to 8.
ports	Specify a pointer to an integer array that contains the SMU port addresses or pin numbers that are assigned to the ports by connect_pin functions. Available *ports* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
statuses	Specify a pointer to an integer array in which to return the measurement statuses. Measurement statuses are returned to array elements that correspond to ports specified in *ports* array.
times	Specify a pointer to a double array in which to return the time stamp data. Time stamp data are returned to array elements that correspond to ports specified in *ports* array.

See also

- "sweep_iv"
- "sweep_miv"

NOTE

To Clear Timer Count

Execute the reset_timestamp function. Then the timer count is not cleared. Execute the force_i or force_v function. Then the timer count will be cleared.

Once the force_i or force_v function is executed, the effects of the reset_timestamp function will be disabled.

sweep_iv

This function performs one of the following sweep measurements:

- staircase sweep measurement (set_iv)
- pulsed sweep measurement (set_piv)
- staircase sweep with pulsed bias measurement (set_iv and set_pbias)

For each sweep step, measurements are made by the channel set to this function. After the sweep measurement is completed, measurement values, primary source values, and synchronous source values for each sweep step are returned to arrays.

If set_sync function is entered, this function performs synchronous sweep measurement using the primary sweep source (set by set_iv or set_piv) and synchronous sweep source (set by set_sync). Also, a pulse source may be used if set_pbias function is entered after the set_iv function.

NOTE

To get the status data

This function does not return status data. Use status_miv function to get the statuses of each measurement points or port_status function to get the status of last measurement point.

Synopsis

```
int sweep_iv(port, mode, range, measure, source, sync)

int port, mode;
double range, measure[n], source[n], sync[n];
```

Arguments

Item	Description
port	Specify measurement channel by using the port address or a pin number that is assigned to the port by the connect_pin function. Available *port* values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.
mode	Specify measurement mode by using the following macros (or values): **V_MEAS (1)** Voltage measurement. **I_MEAS (2)** Current measurement.

Item	Description
range	Measurement ranging mode. Numeric expression. See Table 2-3 on page 2-9 to know the available values.
measure	Specify a pointer to a double array in which to return the measurement results.
source	Specify a pointer to a double array in which to return the sweep source values (which are determined by set_iv or set_piv function). If these values are not necessary, specify NULL (*sync* must also be NULL).
sync	Specify a pointer to a double array in which to return the secondary sweep source values (which are determined by set_sync function). If set_sync function is *not* used, this argument must be NULL. And if these values are not necessary, specify NULL.

Example

```
int nop=11, n=1, ports[1], statuses[11];
double range1, v1, v2, hold1, delay1, comp1, pcomp1, offset;
double ratio, comp2, pcomp2, voltage3, range3, comp3, mrange;
double meas[11], swp1[11], swp2[11], times[11];

   : : : : :

ports[0] = 1;
set_timestamp(1);
reset_timestamp();
set_iv(SMU1, LINEAR_V, range1, v1, v2, nop, hold1, delay1, comp1,
pcomp1, COMP_STOP);
set_sync(SMU2, POS_SYNC, offset, ratio, comp2, pcomp2);
force_v(SMU3, voltage3, range3, comp3);
sweep_iv(SMU1, I_MEAS, mrange, meas, swp1, swp2);
status_miv(n, ports, statuses, times);
```

See Also

- “set_iv”, “set_piv”, “set_sync”, “sweep_miv”, and “status_miv”

sweep_miv

This function performs staircase sweep measurement using up to eight SMU channels. The channels perform measurement sequentially in the order defined in the *ports*.

Before this function, execute force_i or force_v function to set the measurement mode for each channel. The force_i function sets the channel to voltage measurement mode, and force_v sets the channel to current measurement mode.

This function is not available for pulsed sweep measurement (set by set_piv) and staircase sweep with pulsed bias measurement (set by set_iv and set_pbias).

NOTE

To get the status data

This function does not return status data. Use status_miv function to get the statuses of each measurement points or port_status function to get the status of last measurement point.

Synopsis

```
int sweep_miv(n, ports, ranges, meas_vals, source, sync)

int n, ports[n];
double ranges[n], meas_vals[nx], source[x], sync[x];
```

Where *x* is the *number* of steps specified in set_iv function. For double sweep, *x* should be 2*x*.

Arguments

Item	Description
n	Specify how many ports will perform measurement. The size of the arrays *ports*, *ranges*, and the primary index of *meas_vals* must be *n*. *n* can be from 1 to 8.
ports	Specify a pointer to an integer array that contains the measurement channels. This array must have *n* elements. The array should contain the port addresses or pin numbers that are assigned to the port by the connect_pin function. Available values are 1 to 49, SMU1 to SMU8, and values shown in Table 2-2 on page 2-8.

Item	Description
ranges	Specify a pointer to a double array that contains the measurement ranging mode for each channel. This array must have *n* elements. Specify values in *ranges* array in order that corresponds to elements of the *ports* array. See Table 2-3 on page 2-9 to know the available values.
meas_vals	Specify a pointer to a double array in which to return the measured values. This array must have *n*×*number* elements for a single sweep and 2*n*×*number* elements for a double sweep. Where *number* is the number of sweep steps set to the set_iv function. Measurement results are returned to *meas_vals* array in order that corresponds to elements in the *ports* array.
source	Specify a pointer to a double array in which to return the sweep source data (which is determined by set_iv). This array must have *number* elements for a single sweep and 2×*number* elements for a double sweep. Where *number* is the number of sweep steps set to the set_iv function. If these values are not necessary, specify NULL (*sync* must also be NULL).
sync	Specify a pointer to a double array in which to return the secondary sweep source data (which is determined by set_sync function). This array must have *number* elements for a single sweep and 2×*number* elements for a double sweep. Where *number* is the number of sweep steps set to the set_iv function. If set_sync function is *not* used, this argument must be NULL. And if these values are not necessary, specify NULL.

Example

```
int nop=11, n=3, ports[3], statuses[33];
double range1, v1, v2, hold1, delay1, comp1, pcomp1, offset;
double ratio, comp2, pcomp2, voltage3, range3, comp3, mrange;
double ranges[3], meas[33], swp1[11], swp2[11], times[33];

   : : : : :

ports[0] = 1;
ports[1] = 2;
ports[2] = 3;
ranges[0] = 0.0;
ranges[1] = 0.0;
ranges[2] = 0.0;

   : : : : :

set_timestamp(1);
reset_timestamp();
set_iv(SMU1, LINEAR_V, range1, v1, v2, nop, hold1, delay1, comp1,
pcomp1, COMP_STOP);
set_sync(SMU2, POS_SYNC, offset, ratio, comp2, pcomp2);
force_v(SMU3, voltage3, range3, comp3);
sweep_miv(n, ports, ranges, meas, swp1, swp2);
status_miv(n, ports, statuses, times);
```

See Also

- “set_iv”
- “set_sync”
- “sweep_iv”
- “status_miv”

vi_E5270

This function returns the VISA session ID required to use the VXI*plug&play* driver for Agilent E5270B. The driver can be used in a program by using the session ID.

Synopsis

```
ViSession vi_E5270(void)
```

Example

The following example sends the *CAL command to the Agilent E5270B by using the age5270_cmd function that is a function of the Agilent E5270B VXI*plug&play* driver.

```
age5270_cmd(vi_E5270(),"*CAL");
```

3 Programming Examples

This chapter provides programming examples that performs the following measurements using the Agilent E5270B and its TIS library.

- “High Speed Spot Measurement”
- “Multi Channel Spot Measurement”
- “Pulsed Spot Measurement”
- “Staircase Sweep Measurement”
- “Multi Channel Sweep Measurement”
- “Pulsed Sweep Measurement”
- “Staircase Sweep with Pulsed Bias Measurement”
- “Breakdown Voltage Measurement”
- “Leakage Current Measurement”

High Speed Spot Measurement

This section explains an example program that performs the high speed spot measurement. The following program will apply voltage to a device, measure current, calculate the resistance value, and display the measurement result data.

Table 3-1 **High Speed Spot Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif

   int h_pin, l_pin;                                        /* 15 */
   double v_force, comp, current;
   v_force = 0.5;
   comp = 1e-3;
   h_pin = 12;
   l_pin = 16;

   init_system();

   connect_pin(SMU3, h_pin);
   connect_pin(GNDU, l_pin);                                /* 25 */

   force_v(h_pin, v_force, 0, comp);
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 20	Declares variables, and defines the value.
22	Initializes the Agilent E5270B.
24 to 25	Enables measurement ports and assigns terminals to the ports.
27	Applies voltage to the h_pin terminal.

```
   measure_i(h_pin, &current, comp);
                                                        /* 30 */
   disable_port(h_pin);
   disconnect_all();

   printf("I = %8.6f amp (at %4.2f volt)\n", current, v_force);
   printf("R = %6.2f ohm\n", v_force / current);

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );                                            /* 40 */
}
```

Line	Description
29	Performs high speed spot measurement for the h_pin terminal.
31	Sets the specified port to the zero output state.
32	Disables all ports.
34 to 35	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
I = 0.001001 amp (at 0.50 volt)
R = 499.50 ohm
```

Multi Channel Spot Measurement

This section explains an example program that performs the multi channel spot measurement. The following program will apply voltage to a MOSFET, measure drain current and gate current eleven times, obtain time stamp data, and save the measurement results into a file.

Table 3-2 Multi Channel Spot Measurement Example

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int n, drain, gate, source, sub, nop, port[2], samples;
double vd, vg, ig_c, irange, vrange, icomp, current;
double meas[2], range[2], time[2];

vg = 1.5;
vd = 1.5;
vrange = 2.0;
ig_c = 1e-6;
icomp = 50e-3;
irange = 1e-9;
drain = 15;
gate = 16;
source = 17;
sub = 18;
nop =11;
n = 2;
port[0] = drain;
port[1] = gate;
range[0] = icomp;
range[1] = irange;
samples = 100;                                              /* 35 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 35	Declares variables, and defines the value.

```
init_system();                                          /* 37 */

connect_pin(SMU1, drain);
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, sub);

set_timestamp(1);

set_smu_ch(0, PERCH_ADC, FILTER_OFF);                   /* 46 */
set_adc(PERCH_ADC, INTEG_MANUAL, samples, 0);

reset_timestamp();

force_v(sub, 0.0, 2.0, icomp);
force_v(source, 0.0, 2.0, icomp);
force_v(gate, vg, vrange, ig_c);
force_v(drain, vd, vrange, icomp);
```

Line	Description
37	Initializes the Agilent E5270B.
39 to 42	Enables measurement ports and assigns terminals to the ports.
44	Enables time stamp data output.
46 to 47	Selects A/D converter and sets the ADC condition.
49	Enables force_v to clear time stamp.
51 to 54	Applies voltage to device.

```
int i;                                                  /* 56 */
char f_name[] = "data1.txt";
char head1[] = "Index, Id (mA), Time (s), Ig (pA), Time (s)";
char msg1[] = "Measurement in progress.";
char msg2[] = "Measurement and data save completed.";
char c = '\n';

FILE *stream;                                           /* 63 */
if( ( stream = fopen( f_name, "w+" )) == NULL )
  printf( "Data file was not opened\n" );
else
  printf( "%s%c", msg1, c );
  fprintf( stream, "%s%c", head1, c );
  for (i=0; i<nop;i++){                                 /* 69 */
  measure_m(n, port, meas, range, time);
  fprintf(stream, "%2d, %7.4f, %6.4f, %6.4f, %6.4f\n", i+1, mea
s[0]*1000, time[0], meas[1]*1e+12, time[1]);
  }
  printf( "%s%c", msg2, c );

if( fclose( stream ) )
  printf( "Data file was not closed\n"                  /* 76 */

disable_port(0);
disconnect_all();                                       /* 79 */

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
56 to 76	Creates measurement result data file (data1.txt, CSV file).
56 to 61	Declares variables, and sets the values.
63 to 76	Displays error message if the file open function fails, else creates the measurement result data file. And displays error message if the file close function fails.
70	Performs multi channel spot measurement for the drain and gate terminals.
78	Sets all ports to the zero output state.
79	Disables all ports.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Index, Id (mA), Time (s), Ig (pA), Time (s)
 1, 14.2550, 0.0301, 0.7000, 0.0735
 2, 14.2400, 0.1101, 0.5000, 0.1153
 3, 14.2350, 0.1511, 0.3500, 0.1563
 4, 14.2350, 0.1921, 0.3000, 0.1972
 5, 14.2300, 0.2331, 0.3000, 0.2383
 6, 14.2300, 0.2741, 0.2000, 0.2794
 7, 14.2300, 0.3161, 0.1000, 0.3213
 8, 14.2250, 0.3731, 0.0000, 0.3783
 9, 14.2250, 0.4141, 0.1000, 0.4192
10, 14.2250, 0.4550, 0.2000, 0.4602
11, 14.2250, 0.4961, 0.2000, 0.5012
```

Pulsed Spot Measurement

This section explains an example program that performs the pulsed spot measurement. The following program will apply voltage to a MOSFET, measure drain current, and display the measurement result data.

Table 3-3 Pulsed Spot Measurement Example

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int drain, gate, source, sub;
double vd1, vd2, vg, ig_c, irange, vrange, icomp, current;
double width, period, hold;

vg = 1.5;
vd1 = 0.0;
vd2 = 1.5;
vrange = 2.0;
width = 1e-3;
period = 10e-3;
hold = 50e-3;
irange = 100e-3;
ig_c = 5e-3;
icomp = 50e-3;

drain = 15;
gate = 16;
source = 17;
sub = 18;                                                   /* 33 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 33	Declares variables, and defines the value.

```
init_system();                                          /* 35 */
connect_pin(SMU1, drain);
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, sub);
                                                        /* 40 */
force_v(sub, 0.0, vrange, icomp);
force_v(source, 0.0, vrange, icomp);
force_v(gate, vg, vrange, ig_c);
set_pbias(drain, V_SOURCE, vrange, vd1, vd2, width, period,
hold, icomp);
measure_p(drain, I_MEAS, &current, irange);             /* 45 */
disable_port(0);
disconnect_all();

printf("Id = %7.4f mA (at %4.2f V)\n", current*1000, vd2);
printf("Vg = %4.2f V\n", vg);

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );                                            /* 55 */
}
```

Line	Description
35	Initializes the Agilent E5270B.
36 to 39	Enables measurement ports and assigns terminals to the ports.
41 to 43	Applies voltage to device.
44	Sets the pulse bias source for the drain terminal.
45	Performs pulsed spot measurement for the drain terminal.
46	Sets the specified port to the zero output state.
47	Disables all ports.
49 to 50	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Id = 14.2550 mA (at 1.50 V)
Vg = 1.50 V
```

Staircase Sweep Measurement

This section explains an example program that performs the staircase sweep measurement. The following program will performs MOSFET Id-Vg measurement, obtain status information and time stamp data, and display the result data.

Table 3-4 Staircase Sweep Measurement Example

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int drain, gate, source, sub, nop, samples;
int n = 1, port[1], status[11];
double vrange1, vrange2, vd, hold, delay, icomp;
double ig_c, d_p_c, g_p_c;
double irange, id[11], swp1[11], swp2[11], time[11];

vrange1 = 2.0;
vrange2 = 20.0;
irange = 0.0;
vd = 2.5;
nop = 11;
hold = 0.1;
delay = 0.0;
ig_c = 1e-3;
icomp = 50e-3;
d_p_c = 0.1;
g_p_c = 0.002;
drain = 15;
gate = 16;
source = 17;
sub = 18;
samples = 100;                                              /* 36 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 36	Declares variables, and defines the value.

```
init_system();                                          /* 39 */

connect_pin(SMU1, drain);
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, sub);

port[0] = drain;                                        /* 46 */

set_timestamp(1);
set_smu_ch(0, PERCH_ADC, FILTER_OFF);
set_adc(PERCH_ADC, INTEG_MANUAL, samples, 0);

force_v(source, 0.0, vrange1, icomp);                   /* 52 */
reset_timestamp();
force_v(sub, 0.0, vrange1, icomp);

set_iv(drain, LINEAR_V, vrange2, 0.0, vd, nop, hold, delay,
icomp, d_p_c, COMP_STOP);

set_sync(gate, POS_SYNC, 0.0, 1.0, ig_c, g_p_c);

sweep_iv(drain, I_MEAS, irange, id, swp1, swp2);

status_miv(n, port, status, time);                      /* 62 */
```

Line	Description
39	Initializes the Agilent E5270B.
41 to 44	Enables measurement ports and assigns terminals to the ports.
46	Sets port array value.
48	Enables time stamp data output.
49 to 50	Selects A/D converter and sets the ADC condition.
52	Applies voltage to device.
53	Enables force_v to clear time stamp.
54	Applies voltage to device.
56	Sets the voltage sweep source for the drain terminal.
58	Sets the synchronous sweep source for the gate terminal.
60	Performs sweep measurement for the drain terminal.
62	Obtains the sweep status information and time stamp data.

```
disable_port(0);                                        /* 64 */
disconnect_all();

int i;
printf(" Vg (V)  Id (mA)  Status Time (s)\n");
for (i=0; i<nop;i++)
printf(" %4.2f    %7.4f   %d      %6.4f\n", swp2[i], id[i] *
1000, status[i], time[i]);

#ifdef E5270_TIS_H                                      /* 72 */
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
64	Sets the specified port to the zero output state.
65	Disables all ports.
67 to 70	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Vg (V)  Id (mA)  Status Time (s)
0.00     0.0050   0      0.1726
0.25     1.9600   0      0.2265
0.50     4.1000   0      0.2803
0.75     6.4250   0      0.3340
1.00     8.8950   0      0.3877
1.25    11.5150   0      0.4416
1.50    14.2650   0      0.4956
1.75    17.1250   0      0.5497
2.00    20.0850   0      0.6037
2.25    23.1350   0      0.6579
2.50    26.2650   0      0.7119
```

Multi Channel Sweep Measurement

This section explains an example program that performs the multi channel sweep measurement. The following program will performs bipolar transistor Ic-Vbe and Ib-Vbe measurement, calculate hfe value, and save the measurement results into a file.

Table 3-5 **Multi Channel Sweep Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int emitter, base, collector, nop=11, n=2, port[2], samples;
double vrange, vc, iecomp, iccomp, ibcomp, vbstart, vbstop;
double hold, delay, pbcomp;
double swp1[11], swp2[11], meas[22], range[2];

vrange = 20.0;
vc = 3.0;
iecomp = 0.2;
iccomp = 0.2;
ibcomp = 0.005;
vbstart = 0.5;
vbstop = 1.0;
hold = 0.0;
delay = 0.1;
pbcomp = 0.5;
samples = 200;

emitter = 12;
base = 13;
collector = 14;                                             /* 34 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 34	Declares variables, and defines the value.

```
init_system();                                          /* 36 */
connect_pin(SMU1, emitter);
connect_pin(SMU2, base);
connect_pin(SMU4, collector);

port[0] = collector;
port[1] = base;
range[0] = 0.0;
range[1] = 0.0;

set_smu_ch(0, PERCH_ADC, FILTER_OFF);                   /* 46 */
set_adc(PERCH_ADC, INTEG_MANUAL, samples, 0);
force_v(emitter, 0.0, 2.0, iecomp);
force_v(collector, vc, vrange, iccomp);
set_iv(base, LINEAR_V, vrange, vbstart, vbstop, nop, hold,
delay, ibcomp, pbcomp, COMP_STOP);

printf("Measurement in progress...\n");
sweep_miv(n, port, range, meas, swp1, swp2);
                                                        /* 54 */
disable_port(0);
disconnect_all();
```

Line	Description
36	Initializes the Agilent E5270B.
37 to 39	Enables measurement ports and assigns terminals to the ports.
41 to 44	Defines port and range array values.
46 to 47	Selects A/D converter and sets the ADC condition.
48 to 49	Applies voltage to device.
50	Sets the voltage sweep source for the base terminal.
53	Performs multi channel sweep measurement for the collector and base terminals.
55	Sets the specified port to the zero output state.
56	Disables all ports.

```
int i;
char f_name[] = "C:\Agilent\data\data1.txt";
char head1[] = " Vbe (V), Ic (mA), Ib (mA), hfe";     /* 60 */
char msg1[] = "Saving data...";
char msg2[] = "Data save completed.";
char c = '\n';

FILE *stream;                                          /* 65 */
if( ( stream = fopen( f_name, "w+" )) == NULL )
   printf( "Data file was not opened\n" );
else
   printf( "%s%c", msg1, c );
   fprintf( stream, "%s%c", head1, c );
   for (i=0; i<nop;i++)
   fprintf( stream, " %4.2f, %10.6f, %8.6f, %8.4f\n", swp1[i],
meas[i]*1e3, meas[nop+i]*1e3, meas[i]/meas[nop+i]);
   printf( "%s%c", msg2, c );

if( fclose( stream ) )
printf( "Data file was not closed\n" );               /* 76 */

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
58 to 76	Creates result data file (C:\Agilent\data\data1.txt, CSV file).
58 to 63	Declares variables, and sets the values.
65 to 76	Displays error message if the file open function fails, else creates the measurement result data file. And displays error message if the file close function fails.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Vbe (V), Ic (mA), Ib (mA), hfe
0.50,    0.002176, 0.000013, 164.4369
0.55,    0.014675, 0.000083, 176.9992
0.60,    0.102025, 0.000537, 190.0969
0.65,    0.678050, 0.003377, 200.8145
0.70,    4.379500, 0.021385, 204.7931
0.75,   21.095000, 0.108350, 194.6931
0.80,   61.210000, 0.370250, 165.3207
0.85, 104.145000, 1.086000,  95.8978
0.90, 150.850000, 2.487500,  60.6432
0.95, 199.700000, 4.595000,  43.4603
0.00, 9999999999.990000, 9999999999.990000,   1.0000
```

Pulsed Sweep Measurement

This section explains an example program that performs the pulsed sweep measurement. The following program will performs MOSFET Id-Vg measurement and display the measurement result data.

Table 3-6 **Pulsed Sweep Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int drain, gate, source, sub, nop;
double vrange1, vrange2, vd, width, period;
double hold, icomp, ig_c, g_p_c;
double irange, id[11], swp1[11], swp2[11];

vrange1 = 2.0;
vrange2 = 20.0;
irange = 0.0;
vd = 2.5;
nop = 11;
width = 1e-3;
period = 5e-3;
hold = 10e-3;
ig_c = 1e-3;
icomp = 50e-3;
g_p_c = 0.002;

drain = 15;
gate = 16;
source = 17;
sub = 18;                                                   /* 35 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 35	Declares variables, and defines the value.

```
init_system();                                          /* 37 */
connect_pin(SMU1, drain);
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, sub);

force_v(source, 0.0, vrange1, icomp);                   /* 43 */
force_v(sub, 0.0, vrange1, icomp);
set_piv(drain, LINEAR_V, vrange2, 0.0, 0.0, vd, nop, width,
period, hold, icomp, COMP_STOP);
set_sync(gate, POS_SYNC, 0.0, 1.0, ig_c, g_p_c);
sweep_iv(drain, I_MEAS, irange, id, swp1, swp2);

disable_port(0);                                        /* 49 */
disconnect_all();

int i;                                                  /* 52 */
printf(" Vg (V)  Id (mA)\n");
for (i=0; i<nop;i++)
printf(" %4.2f    %7.4f \n", swp2[i], id[i] * 1000);

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
37	Initializes the Agilent E5270B.
38 to 41	Enables measurement ports and assigns terminals to the ports.
43 to 44	Applies voltage to device.
45	Sets the pulsed sweep source for the drain terminal.
46	Sets the synchronous sweep source for the gate terminal.
47	Performs pulsed sweep measurement for the drain terminal.
49	Sets the specified port to the zero output state.
50	Disables all ports.
52 to 55	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Vg (V)   Id (mA)
0.00      0.0050
0.25      1.9600
0.50      4.1000
0.75      6.4250
1.00      8.8950
1.25     11.5150
1.50     14.2650
1.75     17.1250
2.00     20.0850
2.25     23.1350
2.50     26.2650
```

Staircase Sweep with Pulsed Bias Measurement

This section explains an example program that performs the staircase sweep with pulsed bias measurement. The following program will performs MOSFET Id-Vg measurement and display the measurement result data.

Table 3-7 **Staircase Sweep with Pulsed Bias Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int drain, gate, source, sub, nop;
double vrange1, vrange2, vd, vg, width, period;
double hold, delay, icomp, ig_c, pcomp;
double irange, id[11], swp1[11], swp2[11];
vrange1 = 2.0;
vrange2 = 20.0;
irange = 0.0;
vd = 2.5;
vg = 3.0;
nop = 11;
width = 0.001;
period = 0.01;
hold = 0.1;
delay = 0.0;
ig_c = 1e-6;
icomp = 50e-3;
pcomp = 0.07;

drain = 15;
gate = 16;
source = 17;
sub = 18;                                                   /* 37 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 37	Declares variables, and defines the value.

```
init_system();
connect_pin(SMU1, drain);                                    /* 40 */
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, sub);

force_v(source, 0.0, vrange1, icomp);
force_v(sub, 0.0, vrange1, icomp);
set_iv(drain, LINEAR_V, vrange2, 0.0, vd, nop, hold, delay,
icomp, pcomp, COMP_STOP);
set_pbias(gate, V_SOURCE, vrange2, 0.0, vg, width, period, hold,
ig_c);
sweep_iv(drain, I_MEAS, irange, id, swp1, swp2);
                                                             /* 50 */
disable_port(0);
disconnect_all();

int i;

printf(" Vd (V)   Id (mA)\n");
for (i=0; i<nop;i++)
printf(" %4.2f    %7.4f \n", swp1[i], id[i] * 1000);

#ifdef E5270_TIS_H                                           /* 60 */
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
39	Initializes the Agilent E5270B.
40 to 43	Enables measurement ports and assigns terminals to the ports.
45 to 46	Applies voltage to device.
47	Sets the voltage sweep source for the drain terminal.
48	Sets the pulse bias source for the gate terminal.
49	Performs sweep measurement for the drain terminal.
51	Sets the specified port to the zero output state.
52	Disables all ports.
54 to 58	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Vg (V)   Id (mA)
0.00      0.0000
0.25      4.1750
0.50      8.0600
0.75     11.6550
1.00     14.9500
1.25     17.9450
1.50     20.6300
1.75     23.0400
2.00     25.1900
2.25     27.1050
2.50     28.0500
```

Breakdown Voltage Measurement

This section explains an example program that performs the breakdown voltage measurement. The following program will apply quasi pulsed voltage to a bipolar transistor, monitor collector voltage gradient, measure collector voltage when the voltage source reaches it current compliance, and display the measurement result data. The measured voltage is considered as the breakdown voltage for this measurement method.

Table 3-8 **Breakdown Voltage Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int emitter, base, collector, status;
double range, start, stop, current, hold, delay;
double result;

emitter = 12;
base = 13;
collector = 14;
range = 0.0;
start = 0.0;
stop = 200.0;        /* interlock cable must be connected */
current = 15e-3;
hold = 0.1;
delay = 0.1;                                                /* 27 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 27	Declares variables, and defines the value.

```
init_system();
connect_pin(SMU1, emitter);                                    /* 30 */
connect_pin(SMU4, collector);

force_v(emitter, 0.0, 0.0, 0.1);
set_bdv(collector, range, start, stop, current, hold, delay);
measure_bdv( &result, &status, BDV_INTVL_SHORT);

disable_port(0);
disconnect_all();

printf(" Vbd (V)   Status\n");                                 /* 40 */
printf(" %7.4f    %d\n", result, status);

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
29	Initializes the Agilent E5270B.
30 to 31	Enables measurement ports and assigns terminals to the ports.
33	Applies voltage to the emitter terminal.
34	Sets the quasi pulsed voltage source for the collector terminal.
35	Triggers the quasi pulsed voltage source, and performs breakdown voltage measurement for the collector terminal.
37	Sets the specified port to the zero output state.
38	Disables all ports.
40 to 41	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Vbd (V)    Status
57.3850     0
```

Leakage Current Measurement

This section explains an example program that performs the leakage current measurement. The following program will apply quasi pulsed voltage to a MOSFET, monitor drain voltage gradient, measure drain current at the stop voltage, and display the measurement result data.

Table 3-9 **Leakage Current Measurement Example**

```
#include <stdio.h>                                          /* 1 */
#include "E5270_TIS.h"

main()
{
int ret;
#ifdef E5270_TIS_H
   ret = open_E5270( "GPIB0::17::INSTR" , ERR_DETECT_ON,NULL);
   if ( ret != 0 ) {
         printf( "E527x device open failed. Exit program.\n" );
         return( -1 );
   }
#endif                                                      /* 13 */

int drain, gate, source, substrate, status;
double range, vout, comp, start, hold, delay;
double result;

drain = 15;
gate = 16;
source = 17;
substrate = 18;
range = 0.0;
vout = 5.0;
comp = 0.05;
start = -5.0;
hold = 0.01;
delay = 0.01;                                               /* 28 */
```

Line	Description
first 13 lines	Codes defined in the template source file. Establishes the software connection to the Agilent E5270B.
15 to 28	Declares variables, and defines the value.

```
init_system();                                          /* 30 */
connect_pin(SMU1, drain);
connect_pin(SMU2, gate);
connect_pin(SMU4, source);
connect_pin(SMU6, substrate);
                                                        /* 35 */
force_v(substrate, 0.0, 0.0, 0.05);
force_v(source, 0.0, 0.0, 0.05);
force_v(gate, 0.0, 0.0, 0.01);
set_ileak(drain,range, vout, comp, start, hold, delay);
measure_ileak(drain, &result, &status, ILEAK_INTVL_SHORT);

disable_port(0);
disconnect_all();

printf(" Ileak (mA)    Status\n");                      /* 45 */
printf(" %9.6f     %d\n", result*1000, status);

#ifdef E5270_TIS_H
   close_E5270();
#endif
return( 0 );
}
```

Line	Description
30	Initializes the Agilent E5270B.
31 to 34	Enables measurement ports and assigns terminals to the ports.
36 to 38	Applies voltage to device.
39	Sets the quasi pulsed voltage source for the drain terminal.
40	Triggers the quasi pulsed voltage source, and performs leakage current measurement for the drain terminal.
42	Sets the specified port to the zero output state.
43	Disables all ports.
45 to 46	Displays the measurement result data.
last 5 lines	Codes defined in the template source file. Terminates the software connection.

Measurement Result Example

```
Ileak (mA)    Status
11.670000      0
```