Agilent E5260 Series
High Speed Measurement Solutions
Agilent E5270 Series
Precision Measurement Solutions

VXIplug&play Driver User's Guide

Agilent Technologies

Notices

Manual Part Number

E5260-90020

Edition

First Edition, October 2004

Agilent Technologies, Inc.
395 Page Mill Road
Palo Alto, CA 94303 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

This manual describes the installation and reference information of the VXI*plug&play* driver for the Agilent E5260/E5270, and consists of the following chapters:

- Chapter 1, “Installation”

  Describes the installation information of the E5260/E5270 VXI*plug&play* driver.

- Chapter 2, “Driver Functions”

  Describes the reference information of the E5260/E5270 VXI*plug&play* driver.

- Chapter 3, “Programming Examples for Visual Basic Users”

  Provides programming examples using the E5260/E5270 VXI*plug&play* driver on Microsoft Visual Basic environment.

- Chapter 4, “Programming Examples for Visual Basic .NET Users”

  Provides programming examples using the E5260/E5270 VXI*plug&play* driver on Microsoft Visual Basic .NET environment.

- Chapter 5, “Programming Examples for C++ Users”

  Provides programming examples using the E5260/E5270 VXI*plug&play* driver on Microsoft Visual C++ environment.

Contents

Contents

Contents

Contents

1 Installation

This chapter describes the system requirements and installation procedure for the Agilent E5260/E5270 VXI*plug&play* driver.

- “System Requirements”
- “Installing VXIplug&play Driver”

System Requirements

The following system environments are required.

- Operating System

  Microsoft Windows XP Professional, Windows 2000, Windows NT 4.0, or Windows 95. It must be supported by the application development environment.

- Application Development Environment (programming environment)

  Microsoft Visual Basic, Microsoft Visual C++, Borland C++Builder, National Instruments LabWindows or LabVIEW, or Agilent VEE.

- Agilent T&M Programmers Toolkit for Visual Studio .NET

  Agilent W1140A or equivalent. Needed for Visual Basic .NET users.

- GPIB (IEEE 488) Interface and 32-bit VISA I/O Library

  Agilent 82357A USB/GPIB interface, E5810A LAN/GPIB gateway, 82350B GPIB interface, or equivalent. These models include Agilent VISA and SICL I/O libraries.

- Computer and peripherals

  Required specifications depend on the application development environment. See manual of the software. The CD-ROM drive is required to install the VXI*plug&play* driver.

- Minimum disk space

  2 MB for the Agilent E5260/E5270 VXI*plug&play* driver

Installing VXI*plug&play* Driver

The installation flow is shown below. If you have already installed the GPIB (IEEE 488) interface, VISA I/O library, and programming software 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. If you use Microsoft Visual Basic .NET, install the Agilent T&M Programmers Toolkit.

6. Install the Agilent E5260/E5270 VXI*plug&play* driver.

   a. Insert the Agilent E5260 Software CD-ROM or the Agilent E5270 Software CD-ROM to the CD-ROM drive connected to your computer.

   b. Execute \Pnp\E5260.exe or \Pnp\E5270.exe on the CD-ROM.

   The setup program installs the driver. See Table 1-1 for the installed files.

Table 1-1 Agilent E5260/E5270 VXIplug&play Driver Files

File Name [a] [b] [c]	Description
<visa path>\Win*xx*\Age52*x*0\age52*x*0.bas	Driver for Microsoft Visual Basic
<visa path>\Win*xx*\Age52*x*0\age52*x*0.c	Driver source code file
<visa path>\Win*xx*\Age52*x*0\age52*x*0.def	DLL export definition file
<visa path>\Win*xx*\Age52*x*0\age52*x*0.fp	Front panel file
<visa path>\Win*xx*\Age52*x*0\age52*x*0.h	Driver header file
<visa path>\Win*xx*\Age52*x*0\age52*x*0.hlp	On-line help file
<visa path>\Win*xx*\Age52*x*0\readme.txt	Read me file
<visa path>\Win*xx*\bin\age52*x*0_32.dll	Driver DLL file
<visa path>\Win*xx*\include\age52*x*0.h	Driver header file
<visa path>\Win*xx*\lib\bc\age52*x*0.lib	Library for Borland C++Builder
<visa path>\Win*xx*\lib\bc\age52*x*0_32.lib	Library for Borland C++Builder
<visa path>\Win*xx*\lib\msc\age52*x*0.lib	Library for Microsoft C++
<visa path>\Win*xx*\lib\msc\age52*x*0_32.lib	Library for Microsoft C++

a. <visa path> indicates the folder specified when you install the VISA I/O library. In the default setting, <visa path> is \Program Files\Visa. If you use the Agilent IO Library, you can verify <visa path> from the Agilent IO Libraries Installation and Path dialog box. To open the dialog box, click the Agilent IO Libraries Control icon on the Windows task bar, click View Documentation, and click Installation and Path Information.

b. Win*xx* depends on the OS of your computer, Winnt for Windows XP, Windows 2000, or Windows NT, or Win95 for Windows 95.

c. 52*x*0: 5260 for Agilent E5260 driver, 5270 for Agilent E5270 driver.

2 Driver Functions

This chapter is the complete reference of VXI*plug&play* driver for the Agilent E5260/E5270.

- “Function List”
- “Parameters”
- “Status Code”
- “Function Reference”

NOTE

To execute the functions

In this section, the prefix of the function names will be age52x0_ for the functions that are effective for both Agilent E5260 series and E5270B. To enter the function in your program, change the prefix to age5260_ or age5270_ as shown below.

age5260_ : for the Agilent E5260 series

age5270_ : for the Agilent E5270B

NOTE

Additional information

See the on-line help of the VXIplug&play drivers, or open the Age5260.hlp or Age5270.hlp file in the directory that the driver is installed. See “Installing VXIplug&play Driver” on page 1-4.

For measurement functions of the Agilent E5260/E5270, see *Agilent E5260/E5270 Programming Guide*.

Function List

Table 2-1 lists all the driver functions for the Agilent E5260/E5270. You will see a brief description of the functions in the table.

Table 2-1 E5260/E5270 Driver Functions

Category	Function	Description
Initialize	age52x0_init	Initializes the software connection with the E5260/E5270.
Close	age52x0_close	Closes the software connection with the E5260/E5270.
Miscellaneous	age52x0_autoCal	Sets the auto calibration mode
	age5260_setAdc	For E5260 series. Sets the number of samples averaged for the measurement.
	age5270_setAdc	For E5270B. Sets the integration time or number of samples for ADC.
	age52x0_resetTimestamp	Clears the timer count (time stamp data).
Channel setup	age52x0_setSwitch	Sets the channel output switch.
	age52x0_setFilter	Sets the output filter.
	age52x0_setSerRes	Sets the series resistor.
	age5270_setAdcType	For E5270B. Selects the ADC type, high speed or high resolution.
	age52x0_abortMeasure	Aborts the present operation and subsequent command execution.
	age52x0_zeroOutput	Sets the channel output to 0 V.
	age52x0_recoverOutput	Recovers the channel output that is set to 0 V by the age52x0_zeroOut function.
	age5270_asuPath	For E5270B. Controls the connection path of the ASU.
	age5270_asuLed	For E5270B. Enables/disables the connection status indicator (LED) of the ASU.
	age5270_asuRange	For E5270B. Enables/disables 1 pA operation of the ASU.

Category	Function	Description
Spot measurement	age52x0_force	Applies DC current or voltage.
	age52x0_spotMeas	Performs high speed spot measurement.
	age52x0_measureM	Performs spot measurement by multiple channels.
Pulsed spot measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setPbias	Sets the pulsed bias source.
	age52x0_measureP	Performs pulsed spot measurement.
Staircase sweep measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setIv	Sets the sweep source.
	age52x0_setSweepSync	Sets the synchronous sweep source.
	age52x0_stopMode	Sets automatic sweep abort and post sweep output.
	age52x0_sweepIv	Performs sweep measurement by one channel.
	age52x0_sweepMiv	Performs sweep measurement by multiple channels.
Pulsed sweep measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setPiv	Sets the pulsed sweep source.
	age52x0_setSweepSync	Sets the synchronous sweep source.
	age52x0_stopMode	Sets automatic sweep abort.
	age52x0_sweepPiv	Performs pulsed sweep measurement.
Staircase sweep with pulsed bias measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setIv	Sets the sweep source.
	age52x0_setPbias	Sets the pulsed bias source.
	age52x0_setSweepSync	Sets the synchronous sweep source.
	age52x0_stopMode	Sets automatic sweep abort and post sweep output.
	age52x0_sweepPbias	Performs sweep measurement with pulsed bias.

Category	Function	Description
Multi channel sweep measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setIv	Sets the sweep source.
	age52x0_setNthSweep	Sets the synchronous sweep source.
	age52x0_stopMode	Sets automatic sweep abort and post sweep output.
	age52x0_msweepIv	Performs sweep measurement by one measurement channel with multiple sweep sources.
	age52x0_msweepMiv	Performs sweep measurement by multiple measurement channels with multiple sweep sources.
Breakdown voltage measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setBdv	Sets the quasi pulse source.
	age52x0_measureBdv	Performs quasi pulsed spot measurement to measure breakdown voltage.
Leakage current measurement	age52x0_force	Applies DC current or voltage.
	age52x0_setIleak	Sets the quasi pulse source.
	age52x0_measureIleak	Performs quasi pulsed spot measurement to measure leakage current.
Primitive Measurement Functions	age52x0_startMeasure	Specifies measurement mode, and performs measurement.
	age52x0_readData	Reads and returns the source setup data or the data measured by the age52x0_startMeasure function.

Category	Function	Description
Utility	age52x0_reset	Executes the E5260/E5270 reset.
	age52x0_self_test	Executes the E5260/E5270 self-test.
	age52x0_error_query	Queries the E5260/E5270 for error code/message.
	age52x0_error_message	Queries for the driver errors.
	age52x0_revision_query	Queries for the E5260/E5270 firmware/driver revisions.
	age52x0_timeOut	Sets the timeout.
	age52x0_timeOut_Q	Queries for the timeout setting.
	age52x0_errorQueryDetect	Sets the automatic error checking.
	age52x0_errorQueryDetect_Q	Queries for the automatic error checking setting.
	age52x0_dcl	Sends the Device Clear.
	age52x0_readStatusByte_Q	Reads the E5260/E5270 status byte.
	age52x0_opc_Q	Checks the E5260/E5270 operation completion status.
Passthrough Functions	age52x0_cmd	Sends a command.
	age52x0_cmdInt	Sends a command with an integer parameter.
	age52x0_cmdReal	Sends a command with a real parameter.
	age52x0_cmdData_Q	Sends a command to read any data.
	age52x0_cmdString_Q	Sends a command to read string response.
	age52x0_cmdInt16_Q	Sends a command to read 16 bit integer response.
	age52x0_cmdInt16Arr_Q	Sends a command to read 16 bit integer array response.
	age52x0_cmdInt32_Q	Sends a command to read 32 bit integer response.
	age52x0_cmdInt32Arr_Q	Sends a command to read 32 bit integer array response.
	age52x0_cmdReal64_Q	Sends a command to read 64 bit real response.
	age52x0_cmdReal64Arr_Q	Sends a command to read 64 bit real array response.

Parameters

NOTE

Macros

Some functions can use macros to set the parameter values. For details of functions and macros, refer to the help file (Age5260.hlp or Age5270.hlp) in the directory that the driver is installed.

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

- "channel value"
- "range value and ranging mode for Agilent E5260 series"
- "range value and ranging mode for Agilent E5270B"
- "Output voltage, resolution, and compliance by range"
- "Output current, resolution, and compliance by range"

In the following tables, the parameters are put in italics such as *channel*.

Table 2-2 ***channel* value**

Mainframe	*channel*	Description
E5270B	2, 3, 4, 6, 7, 8	HPSMU[a] in the slot specified by *channel*.
	1 to 8	MPSMU in the slot specified by *channel*.
	1 to 8	HRSMU in the slot specified by *channel*.
E5260A	2, 3, 4, 6, 7, 8	HPSMU[a] in the slot specified by *channel*.
	1 to 8	MPSMU in the slot specified by *channel*.
E5262A	1	MPSMU in slot 1.
	2	MPSMU in slot 2.
E5263A	1	MPSMU in slot 1.
	2	HPSMU.

a. HPSMU uses two slots. Then *channel* must be the greater slot number. For example, if it is installed in slot 3 and 4, *channel* must be 4.

Table 2-3 ***range*** **value and ranging mode for Agilent E5260 series**

Voltage or current	Available *range* values [a] [b]	Ranging mode used for output/measurement
both	*range* = 0	Auto ranging
voltage	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 (for HPSMU)	200 V limited auto ranging
current	0 < |*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*| ≤200 mA (for MPSMU)	200 mA limited auto ranging
	100 mA < |*range*| ≤1 A (for HPSMU)	1 A 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 ***range*** **value and ranging mode for Agilent E5270B**

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.

Table 2-5 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				
			E5270B			E5260 series	
			HPSMU	MPSMU	HRSMU	HPSMU	MPSMU
0.5 V	25E-6	0 to ± 0.5	NA	±100E-3	±100E-3	NA	NA
2 V	100E-6	0 to ± 2	±1	±100E-3	±100E-3	±1	±200E-3
5 V	250E-6	0 to ± 5	NA	±100E-3	±100E-3	NA	NA
20 V	1E-3	0 to ± 20	±1	±100E-3	±100E-3	±1	±200E-3
40 V	2E-3	0 to ± 20	±1	±100E-3	±100E-3	±1	±200E-3
		to ± 40	±500E-3	±50E-3	±50E-3	±500E-3	±50E-3
100 V	5E-3	0 to ± 20	±1	±100E-3	±100E-3	±1	±200E-3
		to ± 40	±500E-3	±50E-3	±50E-3	±500E-3	±50E-3
		to ± 100	±125E-3	±20E-3	±20E-3	±125E-3	±20E-3
200 V	10E-3	0 to ± 20	±1	NA	NA	±1	NA
		to ± 40	±500E-3			±500E-3	
		to ± 100	±125E-3			±125E-3	
		to ± 200	±50E-3			±50E-3	

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 2-6 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				
			E5270B			E5260 series	
			HPSMU	MPSMU	HRSMU	HPSMU	MPSMU
1 pA	1E-15	0 to ± 1.15 E-12	NA	NA	±100	NA	NA
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	±200	±100
1 µA	50E-12	0 to ± 1.15E-6	±200	±100	±100	±200	±100
10 µA	500E-12	0 to ± 11.5E-6	±200	±100	±100	±200	±100
100 µA	5E-9	0 to ± 115E-6	±200	±100	±100	±200	±100
1 mA	50E-9	0 to ± 1.15E-3	±200	±100	±100	±200	±100
10 mA	500E-9	0 to ± 11.5E-3	±200	±100	±100	±200	±100
100 mA	5E-6	0 to ± 20E-3	±200	±100	±100	±200	±100
		to ± 50E-3	±200	±40	±40	±200	±40
		to ± 100E-3	±100	±20	±20	±100	±20
		to ± 115E-3	±100	NA	NA	±100	±20
200 mA	10E-6	0 to ± 20E-3	NA			NA	±100
		to ± 50E-3					±40
		to ± 200E-3					±20
1 A	50E-6	0 to ± 50E-3	±200			±200	NA
		to ± 125E-3	±100			±100	
		to ± 500E-3	±40			±40	
		to ± 1	±20			±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.

Status Code

After measurement is performed, the Agilent E5260/E5270 returns a status code to notify you if the measurement has been completed successfully. The status code will be returned with the measurement data by the following functions that perform measurement. Available status values are listed in Table 2-7.

- “age52x0_spotMeas”
- “age52x0_measureM”
- “age52x0_measureP”
- “age52x0_sweepIv”
- “age52x0_sweepMiv”
- “age52x0_msweepIv”
- “age52x0_msweepMiv”
- “age52x0_sweepPiv”
- “age52x0_sweepPbias”
- “age52x0_measureBdv”
- “age52x0_measureIleak”
- “age52x0_readData”

NOTE

If multiple status conditions were found

Sum of the status values will be returned. For example, if an A/D converter overflow occurred, and an SMU was oscillating during the measurements, the returned value is 3 (=1+2).

Table 2-7 **Status Values**

Value	Bit	Description
0	-	No error.
1	1 (LSB)	A/D converter overflowed.
2	2	One or more channels are oscillating.
4	3	Another channel reached its compliance setting.
8	4	This channel reached its compliance setting.
		Normal post-measurement state by age52x0_measureBdv.
16	5	Target value was not found within the search range. (for age52x0_readData)
		Detection time was too long. (for age52x0_measureBdv and age52x0_measureIleak)
32	6	Search measurement was automatically stopped. (for age52x0_readData)
		Output slew rate was too late. (for age52x0_measureBdv and age52x0_measureIleak)

Function Reference

This section describes the functions of VXI*plug&play* driver for the Agilent E5260/E5270. The functions are appeared in alphabetical order.

age52x0_abortMeasure

This function aborts the E5260/E5270's present operation, such as the measurement executed by the age52x0_startMeasure function, the dc bias output by the age52x0_force function, and so on.

Syntax

ViStatus _VI_FUNC age52x0_abortMeasure(ViSession vi);

Parameters

vi	Instrument handle returned from age52x0_init( ).

age5270_asuLed

The Agilent E5260 series does not have this function. This function is available for the Agilent E5270B installed with the high resolution SMU (HRSMU) and the Atto Sense and Switch Unit (ASU).

Disables or enables the connection status indicator (LED) of the ASU. This function is effective for the specified channel.

Syntax

ViStatus _VI_FUNC age5270_asuLed(ViSession vi, ViInt32 channel, ViInt32 mode);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number where the HRSMU has been installed. The ASU must be connected to the HRSMU. 1 to 8. See Table 2-2.
mode	0: Disables the indicator. 1: Enables the indicator. Default setting.

age5270_asuPath

The Agilent E5260 series does not have this function. This function is available for the Agilent E5270B installed with the high resolution SMU (HRSMU) and the Atto Sense and Switch Unit (ASU). This function is not effective when the HIGH VOLTAGE indicator of the Agilent E5270B has been lighted.

Controls the connection path of the ASU. Switches the ASU input resource (HRSMU or the instrument connected to the AUX input) to be connected to the ASU output. This function is effective for the specified channel.

When the Agilent E5270B is turned on, the ASU output will be connected to the SMU connector side, but the HRSMU will not be enabled yet. After this function is executed with *path*=2, the HRSMU specified by *channel* cannot be used. After this function is executed with *path*=1, the HRSMU output will appear on the ASU output. Then the HRSMU output will be 0 V.

Syntax

ViStatus _VI_FUNC age5270_asuPath(ViSession vi, ViInt32 channel, ViInt32 path);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number where the HRSMU has been installed. The ASU must be connected to the HRSMU. 1 to 8. See Table 2-2.
path	Path connected to the ASU output. 1 (the ASU output will be connected to the SMU connector side) or 2 (the ASU output will be connected to the AUX connector side).

NOTE

To use ASU

Before turn the Agilent E5270B on, connect the ASU to the HRSMU properly. The ASU will add the 1 pA range to the HRSMU. If you use other instrument such as the capacitance meter, connect the instrument to the AUX input of the ASU. The ASU provides the input selection function.

Remember that the series resistor in the HRSMU connected to the ASU cannot be used.

age5270_asuRange

The Agilent E5260 series does not have this function. This function is available for the Agilent E5270B installed with the high resolution SMU (HRSMU) and the Atto Sense and Switch Unit (ASU).

Enables or disables the 1 pA range for the auto ranging operation.

Syntax

ViStatus _VI_FUNC age5270_asuRange(ViSession vi, ViInt32 channel, ViInt32 mode);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number where the HRSMU has been installed. The ASU must be connected to the HRSMU. 1 to 8. See Table 2-2.
mode	0: Enables 1 pA range. 1: Disables 1 pA range.

age52x0_autoCal

This function enables or disables the auto calibration function.

Syntax

ViStatus _VI_FUNC age52x0_autoCal(ViSession vi, ViInt32 state);

Parameters

vi	Instrument handle returned from age52x0_init( ).
state	Auto calibration mode. 0 (off) or 1 (on).

age52x0_close

This function terminates the software connection to the instrument and deallocates system resources. It is generally a good programming habit to close the instrument handle when the program is done using the instrument.

Syntax

ViStatus _VI_FUNC age52x0_close(ViSession vi);

Parameters

vi	Instrument handle returned from age52x0_init( ).

age52x0_cmd

This function passes the cmd_str string to the instrument. Must be a NULL terminated C string.

Syntax

ViStatus _VI_FUNC age52x0_cmd(ViSession vi, ViString cmd_str);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).

Example

```
ViSession vi;
ViStatus ret;
ret = age52x0_cmd(vi, "AB");   /* sends the AB command */
```

age52x0_cmdData_Q

This function passes the cmd_str string to the instrument. This entry point will wait for a response which may be any data. You specify the cmd_str and size parameters, and get result[ ].

Syntax

ViStatus _VI_FUNC age52x0_cmdData_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
size	Length of result in bytes. 2 to 32767.
result[ ]	Response from instrument.

age52x0_cmdInt

This function passes the cmd_str string to the instrument. This entry point passes the string in cmd_str followed by a space and then the integer in value. Note that either an Int16 or 32 can be passed as the Int16 will be promoted.

Syntax

ViStatus _VI_FUNC age52x0_cmdInt(ViSession vi, ViString cmd_str, ViInt32 value);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
value	Parameter for command. -2147483647 to 2147483647.

age52x0_cmdInt16Arr_Q

This function passes the cmd_str string to the instrument. This command expects a response that is a definite arbitrary block of 16 bit integers. You specify the cmd_str and size parameters, and get result[ ] and count.

Syntax

ViStatus _VI_FUNC age52x0_cmdInt16Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt16 _VI_FAR result[ ], ViPInt32 count);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
size	Size of result[ ] (number of items in the array). 1 to 2147483647.
result[ ]	Response from instrument.
count	Count of valid items in result[ ]. Returned data.

age52x0_cmdInt16_Q

This function passes the cmd_str string to the instrument. This command expects a response that can be returned as a 16 bit integer.

Syntax

ViStatus _VI_FUNC age52x0_cmdInt16_Q(ViSession vi, ViString cmd_str, ViPInt16 result);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
result	Response from instrument.

age52x0_cmdInt32Arr_Q

This function passes the cmd_str string to the instrument. This command expects a response that is a definite arbitrary block of 32 bit integers. You specify the cmd_str and size parameters, and get result[ ] and count.

Syntax

ViStatus _VI_FUNC age52x0_cmdInt32Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt32 _VI_FAR result[ ], ViPInt32 count);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
size	Size of result[ ] (number of items in the array). 1 to 2147483647.
result[ ]	Response from instrument.
count	Count of valid items in result[ ]. Returned data.

age52x0_cmdInt32_Q

This function passes the cmd_str string to the instrument. This command expects a response that can be returned as a 32 bit integer.

Syntax

ViStatus _VI_FUNC age52x0_cmdInt32_Q(ViSession vi, ViString cmd_str, ViPInt32 result);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
result	Response from instrument.

age52x0_cmdReal

This function passes the cmd_str string to the instrument. This entry point passes the string in cmd_str followed by a space and then the real in value. Note that either an Real32 or 64 can be passed as the Real32 will be promoted.

Syntax

ViStatus _VI_FUNC age52x0_cmdReal(ViSession vi, ViString cmd_str, ViReal64 value);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
value	Parameter for command. -1E+300 to 1E+300.

age52x0_cmdReal64Arr_Q

This function passes the cmd_str string to the instrument. This command expects a response that is a definite arbitrary block of 64 bit real. You specify the cmd_str and size parameters, and get result[ ] and count.

Syntax

ViStatus _VI_FUNC age52x0_cmdReal64Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViReal64 _VI_FAR result[ ], ViPInt32 count);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
size	Size of result[ ] (number of items in the array). 1 to 2147483647.
result[ ]	Response from instrument.
count	Count of valid items in result[ ]. Returned data.

age52x0_cmdReal64_Q

This function passes the cmd_str string to the instrument. This command expects a response that can be returned as a 64 bit real.

Syntax

ViStatus _VI_FUNC age52x0_cmdReal64_Q(ViSession vi, ViString cmd_str, ViPReal64 result);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
result	Response from instrument.

age52x0_cmdString_Q

This function passes the cmd_str string to the instrument. This entry point will wait for a response which must be a string (character data). You specify the cmd_str and size parameters, and get result[ ].

Syntax

ViStatus _VI_FUNC age52x0_cmdString_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
cmd_str	Instrument command (cannot exceed 256 bytes in length).
size	Length of result in bytes. 2 to 32767.
result[ ]	Response from instrument.

age52x0_dcl

This function sends a device clear (DCL) to the instrument.

A device clear will abort the present operation and enable the instrument to accept a new command or query. This is particularly useful in situations where it is not possible to determine the instrument state. In this case, it is customary to send a device clear before issuing a new instrument driver function. The device clear ensures that the instrument will be able to begin processing the new commands.

Syntax

ViStatus _VI_FUNC age52x0_dcl(ViSession vi);

Parameters

vi	Instrument handle returned from age52x0_init( ).

age52x0_error_message

This function translates the error return value from an instrument driver function to a readable string.

Syntax

ViStatus _VI_FUNC age52x0_error_message(ViSession vi, ViStatus error_number, ViChar _VI_FAR message[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
error_number	Error return value from the driver function.
message[ ]	Error message string. Returned data. This is limited to 256 characters.

age52x0_error_query

This function returns the error numbers and corresponding error messages in the error queue of an instrument. See *Agilent E5260/E5270 User's Guide* for a listing of the instrument error numbers and messages. Instrument errors may occur when you places the instrument in a bad state such as sending an invalid sequence of coupled commands. Instrument errors can be detected by polling. Automatic polling can be accomplished by using the age52x0_errorQueryDetect function.

Syntax

ViStatus _VI_FUNC age52x0_error_query(ViSession vi, ViPInt32 error_number, ViChar _VI_FAR error_message[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
error_number	Instrument's error code. Returned data.
error_message[ ]	Instrument's error message. Returned data. This is limited to 256 characters.

age52x0_errorQueryDetect

This function enables or disables automatic instrument error checking. If automatic error checking is enabled then the driver will query the instrument for an error at the end of each function call.

Syntax

ViStatus _VI_FUNC age52x0_errorQueryDetect(ViSession vi, ViBoolean errorQueryDetect);

Parameters

vi	Instrument handle returned from age52x0_init( ).
errorQueryDetect	Error checking enable (VI_TRUE) or disable (VI_FALSE).

age52x0_errorQueryDetect_Q

This function indicates if automatic instrument error detection is enabled or disabled.

Syntax

ViStatus _VI_FUNC age52x0_errorQueryDetect_Q(ViSession vi, ViPBoolean pErrDetect);

Parameters

vi	Instrument handle returned from age52x0_init( ).
pErrDetect	Error checking enable (VI_TRUE) or disable (VI_FALSE).

age52x0_force

This function specifies the dc current/voltage source, and forces the output immediately. To stop the output, use the age52x0_force function with zero output.

Syntax

ViStatus _VI_FUNC age52x0_force(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 value, ViReal64 comp, ViInt32 polarity);

NOTE

range, value, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6.

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Source output mode. 1 (current) or 2 (voltage).
range	Output ranging mode. 0 (auto) or positive value (limited auto).
value	Source output value (in A or V).
comp	Compliance value. (in V or A). It must be voltage for the current source, or current for the voltage source.
polarity	Compliance polarity. 0 (auto) or 1 (manual). If *polarity*=0, the compliance polarity is automatically set to the same polarity as *value*, regardless of the specified *comp* polarity. The compliance polarity is positive if *value*=0. If *polarity*=1, the specified *comp* polarity is kept.

age52x0_init

This function initializes the software connection to the instrument and optionally verifies that instrument is in the system. In addition, it may perform any necessary actions to place the instrument in its reset state.

If the age52x0_init function encounters an error, then the value of the vi output parameter will be VI_NULL.

Syntax

ViStatus _VI_FUNC age52x0_init(ViRsrc InstrDesc, ViBoolean id_query, ViBoolean do_reset, ViPSession vi);

Parameters

InstrDesc	Instrument description. Examples; GPIB0::1::INSTR.
id_query	VI_TRUE (to perform system verification), or VI_FALSE (do not perform system verification).
do_reset	VI_TRUE (to perform reset operation), or VI_FALSE (do not perform reset operation).
vi	Instrument handle. This is VI_NULL if an error occurred during the init.

age52x0_measureBdv

This function triggers quasi-pulsed spot measurement to measure breakdown voltage, and returns breakdown voltage data and measurement status data. Before executing this function, the age52x0_setBdv function must be executed.

Syntax

ViStatus _VI_FUNC age52x0_measureBdv(ViSession vi, ViInt32 interval, ViPReal64 value, ViPInt32 status);

Parameters

vi	Instrument handle returned from age52x0_init( ).
interval	Settling detection interval. 0 (interval short) or 1 (interval long).
value	Breakdown voltage measurement result. Returned data.
status	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead.

NOTE status value after normal measurement

When the measurement channel performs the breakdown voltage measurement normally, the channel reaches its compliance setting. So, the age52x0_measureBdv function returns *status*=8 after normal measurement.

age52x0_measureIleak

This function triggers quasi-pulsed spot measurement to measure leakage current, and returns current measurement data and measurement status data. Before executing this function, the age52x0_setIleak function must be executed.

Syntax

ViStatus _VI_FUNC age52x0_measureIleak(ViSession vi, ViInt32 channel, ViInt32 interval, ViPReal64 value, ViPInt32 status);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
interval	Settling detection interval. 0 (interval short) or 1 (interval long).
value	Leakage current measurement result. Returned data.
status	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead.

age52x0_measureM

This function executes a multi channel spot measurement by the specified units, and returns the measurement result data, measurement status, and time stamp data.

Syntax

ViStatus _VI_FUNC age52x0_measureM(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel[ ]	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Enter 0 to the last element of channel[ ]. For example, if you use two channels, set the array size to 3, specify the channels to the first and second elements, and enter 0 to the third element.
mode[ ]	Measurement mode. 1 (current) or 2 (voltage).
range[ ]	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViReal64 v1 = 3;     /* output voltage */
ViInt32 vmode = 2;   /* voltage output mode */
ViInt32 mch[3];      /* source and measurement channels */
mch[0] = 1;          /* SMU1 for the 1st measurement channel*/
mch[1] = 2;          /* SMU2 for the 2nd measurement channel*/
mch[2] = 0;
ret = age52x0_setSwitch(vi, mch[0], 1);
ret = age52x0_setSwitch(vi, mch[1], 1);
ret = age52x0_force(vi, mch[0], vmode, 0, 0, 0.1, 0);
ret = age52x0_force(vi, mch[1], vmode, 0, v1, 0.1, 0);

ViInt32 mode[2];     /* measurement mode */
mode[0] = 1;         /* current measurement for 1st channel */
mode[1] = 1;         /* current measurement for 2nd channel */
ViReal64 range[2];   /* measurement range */
range[0] = 0;        /* auto ranging for 1st channel */
range[1] = 0;        /* auto ranging for 2nd channel */
ViReal64 md[2];      /* md[0],md[1]: data of 1st,2nd channel */
ViInt32 st[2];       /* st[0],st[1]: status of 1st,2nd channel */
ret = age52x0_measureM(vi, mch, mode, range, &md[0], &st[0], 0);
```

age52x0_measureP

This function executes a pulsed spot measurement by the specified channel, and returns the measurement result data, measurement status, and time stamp data.

Syntax

ViStatus _VI_FUNC age52x0_measureP(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPReal64 value, ViPInt32 status, ViPReal64 time);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
value	Measurement data. Returned data.
status	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead.
time	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead.

age52x0_msweepIv

This function performs sweep measurement, and returns the number of measurement steps, sweep source data, measurement data, measurement status, and time stamp data.

Before executing this function, the sweep source setup function must be executed. To set the primary sweep source, execute the age52x0_setIv function. To set an synchronous sweep source, execute the age52x0_setNthSweep function. Up to seven synchronous sweep sources can be set by executing the age52x0_setNthSweep function for each channel.

Syntax

ViStatus _VI_FUNC age52x0_msweepIv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. 1 to 8. For the HPSMU, set larger slot number. See Table 2-2.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source setup data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 emitter = 1;      /* SMU1 */
ViInt32 base = 2;         /* SMU2 */
ViInt32 collector = 4;    /* SMU4 */
ViReal64 vb1 = 0.25;
ViReal64 vb2 = 0.75;
ViReal64 vc = 3;
ViReal64 ve = 0;
ViReal64 ibcomp = 0.01;
ViReal64 iccomp = 0.1;
ViReal64 iecomp = 0.1;
ViReal64 pcomp = 0;
ViInt32 nop = 11;
ViReal64 hold = 0;
ViReal64 delay = 0;
ViReal64 s_delay = 0;
ViReal64 p_comp = 0;
ViInt32 mode = 1;         /* current measurement */
ViReal64 range = 0 ;      /* auto range */
ViInt32 rep;
ViReal64 sc[11];
ViReal64 md[11];
ViInt32 st[11];
ViReal64 tm[11];

ret = age52x0_setSwitch(vi, emitter, 1);
ret = age52x0_setSwitch(vi, base, 1);
ret = age52x0_setSwitch(vi, collector, 1);

ret = age52x0_resetTimestamp(vi);

ret = age52x0_force(vi, emitter, 2, 0, ve, iecomp, 0);
ret = age52x0_force(vi, collector, 2, 0, vc, iccomp, 0);
ret = age52x0_setIv(vi, base, 1, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp);

ret = age52x0_msweepIv(vi, collector, mode, range, &rep, &sc[0],
&md[0], &st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[n]: Measurement data (current).

st[n]: Status for the md[n] data.

tm[n]: Time stamp data (measurement start time) for the md[n] data.

where, n = 0 to 10 (integer).

age52x0_msweepMiv

This function performs a multi channel sweep measurement using up to eight measurement channels at a time, and returns the number of measurement steps, sweep source data, measurement data, measurement status, and time stamp data.

Before executing this function, the sweep source setup function must be executed. To set the primary sweep source, execute the age52x0_setIv function. To set an synchronous sweep source, execute the age52x0_setNthSweep function. Up to seven synchronous sweep sources can be set by executing the age52x0_setNthSweep function for each channel.

Syntax

ViStatus _VI_FUNC age52x0_msweepMiv(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViPInt32 point, ViReal64 source[ ] , ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel[ ]	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Enter 0 to the last element of channel[ ]. For example, if you use two channels, set the array size to 3, specify the channels to the first and second elements, and enter 0 to the third element.
mode[ ]	Measurement mode. 1 (current) or 2 (voltage).
range[ ]	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 emitter = 1;      /* SMU1 */
ViInt32 base = 2;         /* SMU2 */
ViInt32 collector = 4;    /* SMU4 */
ViReal64 vb1 = 0.25;
ViReal64 vb2 = 0.75;
ViReal64 vc = 3;
ViReal64 ve = 0;
ViReal64 ibcomp = 0.01;
ViReal64 iccomp = 0.1;
ViReal64 iecomp = 0.1;
ViReal64 pcomp = 0.01;
ViInt32 nop = 11;
ViInt32 mch[3];
ViInt32 mode[2];
ViReal64 range[2];
ViInt32 rep;
ViReal64 sc[11];
ViReal64 md[22];
ViInt32 st[22];
ViReal64 tm[22];
mch[0] = collector;
mch[1] = base;
mch[2] = 0;
mode[0] = 1;           /* current measurement */
mode[1] = 1;           /* current measurement */
range[0] = -0.1;       /* 100 mA fixed range */
range[1] = -0.0001;    /* 100 uA fixed range */
ret = age52x0_setSwitch(vi, emitter, 1);
ret = age52x0_setSwitch(vi, base, 1);
ret = age52x0_setSwitch(vi, collector, 1);
ret = age52x0_resetTimestamp(vi);
ret = age52x0_force(vi, emitter, 2, 0, ve, iecomp, 0);
ret = age52x0_force(vi, collector, 2, 0, vc, iccomp, 0);
ret = age52x0_setIv(vi, base, 1, 0, vb1, vb2, nop, 0, 0, 0, ibcomp,
pcomp);
ret = age52x0_msweepMiv(vi, mch, mode, range, &rep, &sc[0],
&md[0], &st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[2*n]: Data (current) measured by the mch[0] channel.

md[2*n+1]: Data (current) measured by the mch[1] channel.

st[2*n]: Status for the md[2*n] data.

st[2*n+1]: Status for the md[2*n+1] data.

tm[2*n]: Time stamp data (measurement start time) for the md[2*n] data.

tm[2*n+1]: Time stamp data (measurement start time) for the md[2*n+1] data.

where, n = 0 to 10 (integer).

age52x0_opc_Q

This function does the *OPC? common command.

Syntax

ViStatus _VI_FUNC age52x0_opc_Q(ViSession vi, ViPBoolean result);

Parameters

vi	Instrument handle returned from age52x0_init( ).
result	*OPC? command execution result. Returned data. VI_TRUE (Operation complete), or VI_FALSE (Operation is pending).

Example

```
ViSession vi;
ViStatus ret;
ViPBoolean result;
ret = age52x0_opc_Q(vi,&result);
```

age52x0_readData

This function reads and returns the source setup data or the data measured by the age52x0_startMeasure function.

Syntax

ViStatus _VI_FUNC age52x0_readData(ViSession vi, ViPInt32 eod, ViPInt32 data_type, ViPReal64 value, ViPInt32 status, ViPInt32 channel);

Parameters

vi	Instrument handle returned from age52x0_init( ).
eod	End of data flag. 0 (not end of data) or 1 (end of data). Returned data.
data_type	Data type of the value. Returned data. 1: Current measurement data. 2: Voltage measurement data. 3: Current output data. 4: Voltage output data. 5: Time stamp data. 6: Invalid data.
value	Measurement data or source setup data. Returned data.
status	Status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead.
channel	Slot number of the slot that installs the SMU used for measurement/output. Returned data. If *value* is regardless of channel settings, –1 is returned.

age52x0_readStatusByte_Q

This function returns the contents of the status byte register.

Syntax

ViStatus _VI_FUNC age52x0_readStatusByte_Q(ViSession vi, ViPInt16 statusByte);

Parameters

vi	Instrument handle returned from age52x0_init( ).
statusByte	The contents of the status byte are returned in this parameter. Returned data.

age52x0_recoverOutput

This function returns the unit to the settings that are stored by the age52x0_zeroOutput function, and clears the stored unit settings.

Syntax

ViStatus _VI_FUNC age52x0_recoverOutput(ViSession vi, ViInt32 channel);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to return the channel settings. See Table 2-2. Set 0 to select all SMUs.

age52x0_reset

This function places the instrument in a default state. Before issuing this function, it may be necessary to send a device clear to ensure that the instrument can execute a reset. A device clear can be issued by invoking age52x0_dcl function.

Syntax

ViStatus _VI_FUNC age52x0_reset(ViSession vi);

Parameters

vi	Instrument handle returned from age52x0_init( ).

age52x0_resetTimestamp

This function clears the count of the timer (time stamp).

Syntax

ViStatus _VI_FUNC age52x0_resetTimestamp(ViSession vi);

Parameters

vi	Instrument handle returned from age52x0_init( ).

age52x0_revision_query

This function returns the driver revision and the instrument firmware revision.

Syntax

ViStatus _VI_FUNC age52x0_revision_query(ViSession vi,
ViChar_VI_FAR driver_rev[ ], ViChar _VI_FAR instr_rev[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
driver_rev[ ]	Instrument driver revision. Returned data. Up to 256 characters.
instr_rev[ ]	Instrument firmware revision. Returned data. Up to 256 characters.

age52x0_self_test

This function causes the instrument to perform a self-test and returns the result of that self-test. This is used to verify that an instrument is operating properly. A failure may indicate a potential hardware problem.

Syntax

ViStatus _VI_FUNC age52x0_self_test(ViSession vi, ViPInt16 test_result,
ViChar_VI_FAR test_message[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
test_result	Numeric result from self-test operation. Returned data. If no error is detected, 0 is returned.
test_message[ ]	Self-test status message.Returned data. Up to 256 characters.

age5260_setAdc

This function sets the number of samples averaged for the measurement.

Syntax

ViStatus _VI_FUNC age5260_setAdc(ViSession vi, ViInt32 mode,ViInt32 value);

Parameters

vi	Instrument handle returned from age5260_init( ).
mode	Averaging mode. 0 (auto), 1 (manual), or 2 (PLC).
value	Coefficient for a reference value to set the number of averaging samples. 1 to 1023. The reference value is the *initial* value for auto mode, 1 sample for manual mode, or 128 samples for PLC mode. where *initial* value is the value automatically defined by the instrument, and you cannot change.

age5270_setAdc

This function sets the integration time or number of samples that is taken or averaged for the measurement. See also “age5270_setAdcType”.

Syntax

ViStatus _VI_FUNC age5270_setAdc(ViSession vi, ViInt32 adc, ViInt32 mode,ViInt32 value, ViInt32 autozero);

Parameters

vi	Instrument handle returned from age5270_init( ).
adc	A/D converter type. 0 (high-speed) or 1 (high-resolution).
mode	Integration/averaging mode. 0 (auto), 1 (manual), or 2 (PLC).
value	Coefficient for a reference value to set the integration time or number of averaging samples. 1 to 1023. The reference value depends on the adc and mode settings: • For high-resolution ADC: The reference value is the *initial* value for auto mode, 80 µs for manual mode, or 1/power line frequency for PLC mode. • For high-speed ADC: The reference value is the *initial* value for auto mode, 1 sample for manual mode, or 128 samples for PLC mode. where *initial* value is the value automatically defined by the instrument, and you cannot change.
autozero	ADC zero function. 0 (off) or 1 (on).

age5270_setAdcType

This function selects the A/D converter type for the measurement channel.

Syntax

ViStatus _VI_FUNC age5270_setAdcType(ViSession vi, ViInt32 channel, ViInt32 adc);

Parameters

vi	Instrument handle returned from age5270_init( ).
channel	Slot number of the slot that installs the SMU used to perform measurement. See Table 2-2. For the HPSMU, set larger slot number. Set 0 to select all SMUs.
adc	A/D converter type. 0 (high-speed) or 1 (high-resolution).

age52x0_setBdv

This function sets the quasi pulse source used to perform breakdown voltage measurement. After the source setup, execute the age52x0_measureBdv function to trigger the measurement.

After the measurement trigger, the quasi pulse source keeps the start voltage during the hold time. After the hold time, the quasi pulse source starts the voltage transition and settling detection. And the source stops the settling detection and keeps the output when the following condition a or b occurs. After the delay time, the measurement channel starts breakdown voltage measurement.

Condition:

a. Quasi-pulse source reaches it current compliance setting.

b. Output voltage slew rate becomes 1/2 of the rate when starting the settling detection.

The condition b means that the quasi-pulse source applies the voltage close to the stop voltage, or the device under test reaches the breakdown condition.

Syntax

ViStatus _VI_FUNC age52x0_setBdv(ViSession vi, ViInt32 channel, ViReal64 range, ViReal64 start, ViReal64 stop, ViReal64 current, ViReal64 hold, ViReal64 delay);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
range	Voltage output ranging mode. 0 (auto) or positive value (limited auto). For the available values, see Table 2-3 and Table 2-5.
start, stop	Start voltage and stop voltage (in V). See Table 2-5. Difference between *start* and *stop* must be 10 V or more.
current	Current compliance (in A). See Table 2-5.
hold	Hold time (in seconds). 0 to 655.35 s, 0.01 s resolution.
delay	Delay time (in seconds). 0 to 6.5535 s, 0.0001 s resolution.

age52x0_setFilter

This function sets the output filter of the specified channel.

Syntax

ViStatus _VI_FUNC age52x0_setFilter(ViSession vi, ViInt32 channel, ViInt32 state);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to change the filter state. See Table 2-2. For the HPSMU, set larger slot number. Set 0 to select all SMUs.
state	0 (off) or 1 (on).

age52x0_setIleak

This function sets the quasi pulse source used to perform leakage current measurement. After the source setup, execute the age52x0_measureIleak function to trigger the measurement.

After the measurement trigger, the quasi pulse source keeps the start voltage during the hold time. After the hold time, the quasi pulse source starts the voltage transition and settling detection. And the source stops the settling detection and keeps the output when the following condition a or b occurs. After the delay time, the measurement channel starts leakage current measurement.

Condition:

a. Quasi-pulse source reaches it current compliance setting.

b. Output voltage slew rate becomes 1/2 of the rate when starting the settling detection.

The condition b means that the quasi-pulse source applies the voltage close to the measurement voltage.

Syntax

ViStatus _VI_FUNC age52x0_setIleak(ViSession vi, ViInt32 channel, ViReal64 range, ViReal64 voltage, ViReal64 compliance, ViReal64 start, ViReal64 hold, ViReal64 delay);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
range	Voltage output ranging mode. 0 (auto) or positive value (limited auto). For the available values, see Table 2-3 and Table 2-5.
start, voltage	Start voltage and measurement voltage (in V). See Table 2-5. Difference between *start* and *voltage* must be 10 V or more.
current	Current compliance (in A). See Table 2-5.
hold	Hold time (in seconds). 0 to 655.35 s, 0.01 s resolution.
delay	Delay time (in seconds). 0 to 6.5535 s, 0.0001 s resolution.

age52x0_setIv

This function specifies staircase sweep source and sets the parameters. The sweep source is used for the staircase sweep measurements and the staircase sweep with pulsed bias measurements.

For the staircase sweep with pulsed bias measurements, the sweep output synchronizes with the pulse output by the age52x0_setPbias function.

Syntax

ViStatus _VI_FUNC age52x0_setIv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 start, ViReal64 stop, ViInt32 point, ViReal64 hold, ViReal64 delay, ViReal64 s_delay, ViReal64 comp, ViReal64 p_comp);

NOTE range, start, stop, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6.

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Source output mode. 1, 2, 3, 4, -1, -2, -3, or -4. For the log sweep mode, *start* and *stop* must be the same polarity. 1: Voltage-Single-Linear sweep 2: Voltage-Single-Log sweep 3: Voltage-Double-Linear sweep 4: Voltage-Double-Log sweep -1: Current-Single-Linear sweep (only for SMU) -2: Current-Single-Log sweep (only for SMU) -3: Current-Double-Linear sweep (only for SMU) -4: Current-Double-Log sweep (only for SMU)
range	Output ranging mode. 0 (auto) or positive value (limited auto).
start	Sweep start value (in A or V).
stop	Sweep stop value (in A or V).
point	Number of sweep steps. 1 to 1001.
hold	Hold time. 0 to 655.35 seconds, in 0.01 seconds resolution.
delay	Delay time. 0 to 65.535 seconds, in 0.0001 seconds resolution.
s_delay	Step delay time. 0 to 1.0 seconds, in 0.0001 seconds resolution.

comp	Compliance value (in V or A). It must be voltage for the current sweep source, or current for the voltage sweep source. Compliance polarity is automatically set to the same polarity as the output value, regardless of the specified *comp* polarity. The compliance polarity is positive if the output value is 0.
p_comp	Power compliance. Available values are listed below. If you enter the other value, the power compliance is not set. 0.001 to 4.0 VA (for MPSMU) 0.001 to 20.0 VA (for HPSMU) Setting resolution: 0.001 VA

age52x0_setNthSweep

This function sets the synchronous sweep source for the multi channel sweep measurements. Up to seven synchronous sweep sources can be set by entering this function for each channel. The source output is the staircase sweep, and synchronizes with the primary sweep source output. The age52x0_setIv function must be executed before this function.

To perform the multi channel sweep measurements, execute the age52x0_setIv function to set the primary sweep source (first sweep source), and execute the age52x0_msweepIv or age52x0_msweepMiv function to start measurement. The age52x0_msweepMiv function allows to use multiple measurement channels.

Syntax

ViStatus _VI_FUNC age52x0_setNthSweep(ViSession vi, ViInt32 n, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 start, ViReal64 stop, ViReal64 comp, ViReal64 p_comp);

NOTE

range, start, stop, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6.

Sweep type, linear or log, is set by the age52x0_setIv function. If the function sets the log sweep, *start* and *stop* must be the same polarity.

Parameters

vi	Instrument handle returned from age52x0_init( ).
n	Sweep source ID. 2 for the second sweep source (first synchronous sweep source), 3 for the third sweep source (second synchronous sweep source), ..., or 8 for the eighth sweep source (seventh synchronous sweep source).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Source output mode. 1 (current) or 2 (voltage).
range	Output ranging mode. 0 (auto) or positive value (limited auto).
start	Sweep start value (in A or V).
stop	Sweep stop value (in A or V).
comp	Compliance value (in V or A). It must be voltage for the current sweep source, or current for the voltage sweep source. Compliance polarity is automatically set to the same polarity as the output value, regardless of the specified *comp* polarity. The compliance polarity is positive if the output value is 0.

p_comp	Power compliance. Available values are listed below. If you enter the other value, the power compliance is not set. 0.001 to 4.0 VA, 0.001 VA resolution (for MPSMU) 0.001 to 20.0 VA, 0.001 VA resolution (for HPSMU)

age52x0_setPbias

This function specifies pulse source and sets the parameters. The pulse source is used for the pulsed spot measurements and the staircase sweep with pulsed bias measurements. For the staircase sweep with pulsed bias measurements, the pulse output synchronizes with the staircase sweep output by the age52x0_setIv function.

Measurement channel always uses the high-speed A/D converter, and performs measurement so that the pulse width and pulse period are kept. The integration time is automatically set by the Agilent E5260/E5270, and you cannot change. The age52x0_setAdc and age52x0_setAdcType settings are ignored. Also the timing parameters of the age52x0_setIv function are also ignored.

Syntax

ViStatus _VI_FUNC age52x0_setPbias(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 base, ViReal64 peak, ViReal64 width, ViReal64 period, ViReal64 hold, ViReal64 comp);

NOTE range, base, peak, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6.

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Pulse output mode. 1 (current), or 2 (voltage). For the current output, *base* and *peak* must be the same polarity.
range	Output ranging mode. 0 (auto) or positive value (limited auto).
base	Pulse base value (in A or V).
peak	Pulse peak value (in A or V).
width	Pulse width (in seconds). 0.0005 to 2.0 s. 0.0001 s resolution.
period	Pulse period (in seconds). 0.005 to 5.0 s. 0.0001 s resolution. • *period* ≥ *width* +2 msec (for *width* ≤100 ms) • *period* ≥ *width* +10 msec (for *width* > 100 ms) If you set *period*=0, the E5260/E5270 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).
hold	Hold time (in seconds). 0.0 to 655.35 sec. 0.01 sec resolution.

comp	Compliance value (in V or A). It must be voltage for the current source, or current for the voltage source. Compliance polarity is automatically set to the same polarity as the output value, regardless of the specified *comp* polarity. The compliance polarity is positive if the output value is 0.

age52x0_setPiv

This function specifies pulsed sweep source and sets the parameters. The pulsed sweep source is used for the pulsed sweep measurements.

Measurement channel always uses the high-speed A/D converter, and performs measurement so that the pulse width and pulse period are kept. The integration time is automatically set by the Agilent E5260/E5270, and you cannot change. The age52x0_setAdc and age52x0_setAdcType settings are ignored. Also the timing parameters of the age52x0_setIv function are also ignored.

Syntax

ViStatus _VI_FUNC age52x0_setPiv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 base, ViReal64 start, ViReal64 stop, ViInt32 point, ViReal64 hold, ViReal64 width, ViReal64 period, ViReal64 comp);

NOTE

range, base, start, stop, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6.

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Pulse output mode. 1, 3, -1, or -3. For the current output, *base*, *start*, and *stop* must be the same polarity. 1: Voltage-Single-Linear sweep 3: Voltage-Double-Linear sweep -1: Current-Single-Linear sweep -3: Current-Double-Linear sweep
range	Output ranging mode. 0 (auto) or positive value (limited auto).
base	Pulse sweep base value (in A or V).
start	Pulse sweep start value (in A or V).
stop	Pulse sweep stop value (in A or V).
point	Number of sweep steps. 1 to 1001.
hold	Hold time (in seconds). 0.0 to 655.35 sec. 0.01 sec resolution.
width	Pulse width (in seconds). 0.0005 to 2.0 s. 0.0001 s resolution.

period	Pulse period (in seconds). 0.005 to 5.0 s. 0.0001 s resolution. • *period* ≥ *width* +2 msec (for *width* ≤100 ms) • *period* ≥ *width* +10 msec (for *width* > 100 ms) If you set *period*=0, the E5260/E5270 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).
comp	Compliance value (in V or A). It must be voltage for the current sweep source, or current for the voltage sweep source. Compliance polarity is automatically set to the same polarity as the output value, regardless of the specified *comp* polarity. The compliance polarity is positive if the output value is 0.

age52x0_setSerRes

This function sets the series resistor of the specified channel.

Syntax

ViStatus _VI_FUNC age52x0_setSerRes(ViSession vi, ViInt32 channel, ViInt32 state);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Set 0 to select all SMUs.
state	0 (disconnects resistor) or 1 (connects resistor).

age52x0_setSweepSync

This function specifies synchronous sweep source and sets the parameters. The synchronous sweep source will be the additional staircase sweep source for the staircase sweep measurements, the pulsed sweep measurements, or the staircase sweep with pulsed bias measurements. The age52x0_setIv or age52x0_setPiv function must be executed before this function.

For the staircase sweep measurements, the output synchronizes with the staircase sweep output by the age52x0_setIv function.

For the pulsed sweep measurements, the output synchronizes with the pulsed sweep output by the age52x0_setPiv function.

For the staircase sweep with pulsed bias measurements, the output synchronizes the staircase sweep output by the age52x0_setIv function and the pulse output by the age52x0_setPbias function.

Syntax

ViStatus _VI_FUNC age52x0_setSweepSync(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 start, ViReal64 stop, ViReal64 comp, ViReal64 p_comp);

NOTE

range, start, stop, comp parameters

Available values depend on the unit. See Table 2-3 and Table 2-5 or Table 2-6

Sweep type, linear or log, is set by the age52x0_setIv function. If the function sets the log sweep, *start* and *stop* must be the same polarity.

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Source output mode. 1 (current) or 2 (voltage). Set 1 if the age52x0_setIv or age52x0_setPiv function sets the current output mode. Or, set 2 if the function sets the voltage output mode.
range	Output ranging mode. 0 (auto) or positive value (limited auto).
start	Sweep start value (in A or V).
stop	Sweep stop value (in A or V).

comp	Compliance value (in V or A). It must be voltage for the current sweep source, or current for the voltage sweep source. Compliance polarity is automatically set to the same polarity as the output value, regardless of the specified *comp* polarity. The compliance polarity is positive if the output value is 0.
p_comp	Power compliance. 0.001 to 4.0 VA (for MPSMU), or 0.001 to 20.0 VA (for HPSMU) in 0.001 VA resolution. If you enter the other value, the power compliance is not set.

age52x0_setSwitch

This function sets the output switch of the specified channel.

Syntax

ViStatus _VI_FUNC age52x0_setSwitch(ViSession vi, ViInt32 channel, ViInt32 state);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Set 0 to select all channels.
state	Output switch setting. 0 (off) or 1 (on).

age52x0_spotMeas

This function executes a high speed spot measurement by the specified channel, and returns the measurement result data, measurement status, and time stamp data.

Syntax

ViStatus _VI_FUNC age52x0_spotMeas(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPReal64 value, ViPInt32 status, ViPReal64 time);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
value	Measurement data. Returned data.
status	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead.
time	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead.

age52x0_startMeasure

This function starts the specified measurement by the specified channels. You can read the measured data by using the age52x0_readData function. The measurement data is entered to the E5260/E5270 output buffer in the measurement order. If you want to abort the measurement, use the age52x0_abortMeasure function.

Syntax

ViStatus _VI_FUNC age52x0_startMeasure(ViSession vi, ViInt32 meas_type, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViInt32 source, ViInt32 timestamp);

Parameters

vi	Instrument handle returned from age52x0_init( ).
meas_type	Measurement type. 1: multi spot 2: staircase sweep 3: pulse spot 4: pulse sweep 5: sweep with pulsed bias 9: quasi pulsed spot 14: linear search 15: binary search 16: multi channel sweep
channel[ ]	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Enter 0 to the last element of channel[ ]. For example, if you use two channels, set the array size to 3, specify the channels to the first and second elements, and enter 0 to the third element. For *meas_type*=1, 2, or 16, up to eight measurement channels can be set. For *meas_type*=3, 4, 5, or 9, only one measurement channel can be set. For *meas_type*=14 or 15, set 0 (NULL pointer) instead of channel [ ].
mode[ ]	Measurement mode. 1 (current) or 2 (voltage).
range[ ]	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.

source — Enables or disables source data output. 0 (disable) or 1 (enable).

For *meas_type*=9, 14, and 15, enter 0 (zero). Source data output is not available for these measurement types.

timestamp — Enables or disables time stamp data output. Time stamp data is the measurement start time. 0 (disable) or 1 (enable).

For *meas_type*=9, 14, and 15, enter 0 (zero). Time stamp data output is not available for these measurement types.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 mch[3];      /* channel */
mch[0] = 1;          /* SMU1 for the 1st measurement channel*/
mch[1] = 2;          /* SMU2 for the 2nd measurement channel*/
mch[2] = 0;

ret = age52x0_setSwitch(vi, mch[0], 1);
ret = age52x0_setSwitch(vi, mch[1], 1);

ViInt32 om = 2;         /* output mode: voltage */
ViReal64 or = 0;        /* output range: auto */
ViReal64 v1 = 0;        /* base voltage */
ViReal64 v2 = 1.5;      /* peak voltage */
ViReal64 tw = 0.001;    /* width */
ViReal64 tp = 0.01;     /* period */
ViReal64 th = 0;        /* hold time */
ViReal64 ic = 0.01;     /* current compliance */

ret= age52x0_setPbias(vi, mch[0], om, or, v1, v2, tw, tp, th, ic);
ret= age52x0_force(vi, mch[1], om, or, v1, ic, 0);

ViInt32 type = 3;    /* pulsed spot measurement */
ViInt32 mode[2];     /* measurement mode */
ViReal64 range[2];   /* measurement range */
mode[0] = 1;         /* current for 1st measurement channel */
mode[1] = 1;         /* current for 2nd measurement channel */
range[0] = 0;        /* auto for 1st measurement channel */
range[1] = 0;        /* auto for 2nd measurement channel */

ret = age52x0_startMeasure(vi, type, mch, mode, range, 0, 0);

ViInt32 eod;         /* eod */
ViInt32 dtype;       /* data type */
ViReal64 md;         /* measurement value */
ViInt32 st;          /* measurement status */
ViInt32 ch;          /* channel */

ret = age52x0_readData(vi, &eod, &dtype, &md, &st, &ch);
printf("I1 = %9.6f mA \n", md * 1000);

ret = age52x0_readData(vi, &eod, &dtype, &md, &st, &ch);
printf("I2 = %9.6f mA \n", md * 1000);
```

age52x0_stopMode

This function enables or disables the automatic sweep abort function, and specifies the source output value after sweep measurement. This function is available for the staircase sweep, pulsed sweep, staircase sweep with pulsed bias, and multi channel sweep measurements.

This function automatically stops sweep if a source channel reaches its compliance, a measurement value exceeds the specified measurement range, or an SMU oscillates.

Syntax

ViStatus _VI_FUNC age52x0_stopMode(ViSession vi, ViInt32 stop, ViInt32 last_mode);

Parameters

vi	Instrument handle returned from age52x0_init( ).
stop	Enables or disables the automatic sweep abort function. 0 (disable) or 1 (enable).
last_mode	Sweep source output value after normal sweep end. 1 (sweep start value) or 2 (sweep stop value). If the sweep measurement is stopped by the automatic sweep abort function, power compliance, or AB command, the sweep source applies the sweep start value. After the pulsed sweep measurement, the pulse sweep source always applies the pulse base value.

age52x0_sweepIv

This function executes a staircase sweep measurement by the specified channel, and returns the number of measurement steps, sweep source data, measurement data, measurement status, and time stamp data.

Before executing this function, set the sweep source setup by using the age52x0_setIv function. If you want to use the synchronous sweep source, execute the age52x0_setSweepSync function.

Syntax

ViStatus _VI_FUNC age52x0_sweepIv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 sch = 1;       /* SMU1 for sweep channel */
ViInt32 mch = 2;       /* SMU2 for measurement channel */
ViInt32 sm = 1;        /* sweep mode: voltage-single-linear */
ViInt32 om = 2;        /* output mode: voltage */
ViReal64 or = 0;       /* output range: auto */
ViReal64 v1 = 0;       /* start voltage */
ViReal64 v2 = 1.5;     /* stop voltage */
ViInt32 pts = 11;      /* point */
ViReal64 th = 0.01;    /* hold time */
ViReal64 td = 0.001;   /* delay time and step delay time */
ViReal64 icomp = 0.1; /* current compliance */
ViReal64 pcomp = 0.2; /* power compliance */
ViInt32 mm = 1;        /* measurement mode: current */
ViReal64 mr = 0;       /* measurement range: auto */
ViInt32 mpts;          /* number of measurement steps */
ViReal64 sc[11];       /* source data */
ViReal64 md[11];       /* measurement data */
ViInt32 st[11];        /* status */
ViInt32 tm[11];        /* time stamp data */
ret = age52x0_setSwitch(vi, sch, 1);
ret = age52x0_setSwitch(vi, mch, 1);
ret = age52x0_resetTimestamp(vi);
ret = age52x0_force(vi, mch, om, or, v1, icomp, 0);
ret = age52x0_setIv(vi, sch, sm, or, v1, v2, pts, th, td, td,
icomp, pcomp);
ret = age52x0_sweepIv(vi, mch, mm, mr, &mpts, &sc[0], &md[0],
&st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[n]: Measurement data (current).

st[n]: Status for the md[n] data.

tm[n]: Time stamp data (measurement start time) for the md[n] data.

where, n = 0 to 10 (integer).

age52x0_sweepMiv

This function executes a multi channel sweep measurement by the specified channels, and returns the number of measurement steps, sweep source data, measurement data, measurement status, and time stamp data.

Before executing this function, set the sweep source setup by using the age52x0_setIv function. If you want to use the synchronous sweep source, execute the age52x0_setSweepSync function.

Syntax

ViStatus _VI_FUNC age52x0_sweepMiv(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel[ ]	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number. Enter 0 to the last element of channel[ ]. For example, if you use two channels, set the array size to 3, specify the channels to the first and second elements, and enter 0 to the third element.
mode[ ]	Measurement mode. 1 (current) or 2 (voltage).
range[ ]	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 mch[3];       /* measurement channels */
mch[0] = 1;
mch[1] = 2;
mch[2] = 0;
ret = age52x0_setSwitch(vi, mch[0], 1);
ret = age52x0_setSwitch(vi, mch[1], 1);

ViInt32 om = 2;    /* output mode: voltage */
ViInt32 sm = 1;    /* sweep mode: voltage-single-linear mode */
ViReal64 or = 0;   /* output range: auto */
ViReal64 v1 = 0;      /* start voltage */
ViReal64 v2 = 1.5;    /* stop voltage */
ViInt32 pts = 11;     /* point */
ViReal64 th = 0.01;   /* hold time */
ViReal64 td = 0.001;  /* delay time */
ViReal64 ts = 0.001;  /* step delay time */
ViReal64 icomp = 0.1; /* current compliance */
ViReal64 pcomp = 0.2; /* power compliance */
ret = age52x0_resetTimestamp(vi);
ret = age52x0_force(vi, mch[0], om, or, v1, icomp, 0);
ret = age52x0_setIv(vi, mch[1], sm, or, v1, v2, pts, th, td, ts,
icomp, pcomp);

ViInt32 mm[2];        /* measurement mode */
ViReal64 mr[2];       /* measurement range */
mm[0] = 1;            /* current mode for mch[0] */
mm[1] = 1;            /* current mode for mch[1] */
mr[0] = 0;            /* auto range for mch[0] */
mr[1] = 0;            /* auto range for mch[1] */
ViInt32 mpts;         /* number of measurement steps */
ViReal64 sc[11];      /* source data */
ViReal64 md[22];      /* measurement data */
ViInt32 st[22];       /* status */
ViInt32 tm[22];       /* time stamp data */
ret = age52x0_sweepMiv(vi, mch, mm, mr, &mpts, &sc[0], &md[0],
&st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[2*n]: Data (current) measured by the mch[0] channel.

md[2*n+1]: Data (current) measured by the mch[1] channel.

st[2*n]: Status for the md[2*n] data.

st[2*n+1]: Status for the md[2*n+1] data.

tm[n]: Time stamp data (measurement start time) for the md[n] data.

tm[2*n+1]: Time stamp data (measurement start time) for the md[2*n+1] data.

where, n = 0 to 10 (integer).

age52x0_sweepPbias

This function executes a staircase sweep with pulsed bias measurement by the specified channel, and returns the number of measurement steps, sweep source data, measurement data, measurement status, and time stamp data. Before executing this function, set the sweep source setup and pulsed bias setup by using the age52x0_setIv function and the age52x0_setPbias function. If you want to use the synchronous sweep source, execute the age52x0_setSweepSync function.

Syntax

ViStatus _VI_FUNC age52x0_sweepPbias(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 pch = 1;        /* SMU1 for pulse source channel */
ViInt32 om = 2;         /* output mode: voltage */
ViReal64 or = 0;        /* output range: auto */
ViReal64 th = 0;        /* hold time */
ViReal64 tw = 0.001;    /* pulse width */
ViReal64 tp = 0.01;     /* pulse period */
ViReal64 v1 = 0;        /* pulse base voltage */
ViReal64 v2 = 1.5;      /* pulse peak voltage */
ViReal64 ic = 0.05;     /* pulse source current compliance */

ret = age52x0_setSwitch(vi, pch, 1);
ret = age52x0_setPbias(vi, pch, om, or, v1, v2, tw, tp, th, ic);

ViInt32 sch = 2;       /* SMU2 for sweep source channel */
ViInt32 sm = 1;        /* sweep mode: voltage-single-linear */
ViInt32 pts = 11;      /* number of sweep steps */
ViReal64 td = 0;       /* delay time */
ViReal64 ts = 0;       /* step delay time */
ViReal64 s1 = 0;       /* sweep start voltage */
ViReal64 s2 = 3;       /* sweep stop voltage */
ViReal64 icomp = 0.1; /* sweep source current compliance */
ViReal64 pcomp = 0.5; /* sweep source power compliance */

ret = age52x0_setSwitch(vi, sch, 1);
ret = age52x0_setIv(vi, sch, sm, or, s1, s2, pts, th, td, ts,
icomp, pcomp);

ViInt32 mm = 1;        /* measurement mode: current */
ViReal64 mr = 0;       /* measurement range: auto */
ViInt32 mpts;          /* number of measurement steps */
ViReal64 sc[11];       /* source data */
ViReal64 md[11];       /* measurement data */
ViInt32 st[11];        /* status */
ViInt32 tm[11];        /* time stamp data */
ret = age52x0_resetTimestamp(vi);

ret = age52x0_sweepPbias(vi, sch, mm, mr, &mpts, &sc[0], &md[0],
&st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[n]: Measurement data (current).

st[n]: Status for the md[n] data.

tm[n]: Time stamp data (measurement start time) for the md[n] data.

where, n = 0 to 10 (integer).

age52x0_sweepPiv

This function executes a pulsed sweep measurement by the specified channel, and returns the number of measurement steps, sweep source data, measurement value, measurement status, and time stamp data.

Before executing this function, set the pulsed sweep source setup by using the age52x0_setPiv function. If you want to use the synchronous sweep source, execute the age52x0_setSweepSync function.

Syntax

ViStatus_VI_FUNC age52x0_sweepPiv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ], ViReal64 time[ ]);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the slot that installs the SMU to be used. See Table 2-2. For the HPSMU, set larger slot number.
mode	Measurement mode. 1 (current) or 2 (voltage).
range	Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-3 or Table 2-4.
point	Number of measurement steps. Returned data.
source[ ]	Sweep source setup data. Returned data. To disable the source data output, set 0 (NULL pointer) instead of array.
value[ ]	Measurement data. Returned data.
status[ ]	Measurement status. Returned data. See “Status Code” on page 2-13. To disable the status data output, set 0 (NULL pointer) instead of array.
time[ ]	Time stamp data (measurement start time). Returned data. To disable the time stamp data output, set 0 (NULL pointer) instead of array.

Example

```
ViSession vi;
ViStatus ret;
ViInt32 pch = 1;      /* SMU1 for pulse sweep source */

ret = age52x0_setSwitch(vi, pch, 1);

ViInt32 sm = 1;       /* sweep mode: voltage-single-linear mode */
ViReal64 or = 0;      /* output range: auto */
ViReal64 v0 = 0;      /* pulse base voltage */
ViReal64 v1 = 0;      /* pulse sweep start voltage */
ViReal64 v2 = 10;     /* pulse sweep stop voltage */
ViInt32 pts = 11;     /* number of sweep steps */
ViReal64 th = 0;      /* hold time */
ViReal64 tw = 0.001; /* pulse width */
ViReal64 tp = 0.01;  /* pulse period */
ViReal64 ic = 0.05;  /* sweep source current compliance */

ret = age52x0_setPiv(vi, pch, sm, or, v0, v1, v2, pts, th, tw, tp,
ic);

ViInt32 mm = 1;       /* measurement mode: current */
ViReal64 mr = 0;      /* measurement range: auto */
ViInt32 mpts;         /* number of measurement steps */
ViReal64 sc[11];      /* source data */
ViReal64 md[11];      /* measurement data */
ViInt32 st[11];       /* status */
ViInt32 tm[11];        /* time stamp data */
ret = age52x0_resetTimestamp(vi);

ret = age52x0_sweepPiv(vi, pch, mm, mr, &mpts, &sc[0], &md[0],
&st[0], &tm[0]);
```

For the above example, the array variables sc[], md[], st[], and tm[] will contain the following data.

sc[n]: Sweep source setup data (voltage).

md[n]: Measurement data (current).

st[n]: Status for the md[n] data.

tm[n]: Time stamp data (measurement start time) for the md[n] data.

where, n = 0 to 10 (integer).

age52x0_timeOut

This function sets a minimum timeout value for driver I/O transactions in milliseconds. The default timeout period is 5 seconds.

Syntax

ViStatus_VI_FUNC age52x0_timeOut(ViSession vi, ViInt32 timeOut);

Parameters

vi	Instrument handle returned from age52x0_init( ).
timeOut	I/O timeout value for all functions in the driver. in milliseconds. 0 to 2147483647.

age52x0_timeOut_Q

This function returns the timeout value for driver I/O transactions in milliseconds.

Syntax

ViStatus_VI_FUNC age52x0_timeOut_Q(ViSession vi, ViPInt32 pTimeOut);

Parameters

vi	Instrument handle returned from age52x0_init( ).
pTimeOut	Minimum timeout period that the driver can be set to, in milliseconds. Returned data.

age52x0_zeroOutput

This function stores the measurement setup of the units, and sets the units to 0 V output. To recover the setup, execute age52x0_recoverOutput function.

Syntax

ViStatus_VI_FUNC age52x0_zetoOutput(ViSession vi, ViInt32 channel);

Parameters

vi	Instrument handle returned from age52x0_init( ).
channel	Slot number of the SMU to set to the zero output. Set 0 to select all SMUs. See Table 2-2.

3 Programming Examples for Visual Basic Users

This chapter provides programming examples to perform the following measurements using the Agilent E5260/E5270 and the Agilent E5260/E5270 VXI*plug&play* driver.

- "Programming Basics"
- "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"

NOTE

About Program Code

Programming examples are provided as subprograms that can be run with the project template shown in Table 3-1. To execute the program, insert the subprograms instead of the perform_meas subprogram in the template.

NOTE

To Start Program

If you create the measurement program by modifying the example code shown in Table 3-1, the program can be run by clicking the Run button on the Visual Basic main window. After that, a message box will appear. Then click OK to continue.

NOTE

For the Agilent E5260 Users

The example program code uses the Agilent E5270 VXI*plug&play* driver. So the modification is required for the Agilent E5260.

At first, delete age5270_asu, age5270_asuLed, and age5270_setAdcType. There is no replaceable function for them. Second, change the prefix of the function name to age5260_. And correct the parameter values for some functions. Available values are different for each SMU. See "Parameters" on page 2-7.

Finally, correct the syntax of the age5260_setAdc function. See "age5260_setAdc" on page 2-38.

Programming Basics

This section provides the basic information for programming using the Agilent E5260/E5270 VXI*plug&play* driver.

- "To Create Your Project Template"
- "To Create Measurement Program"

To Create Your Project Template

This section explains how to create a project template by using Microsoft Visual Basic. Before starting programming, create your project template, and keep it as your reference. It will remove the conventional task in the future programming.

Step 1. Connect instrument (e.g. Agilent E5270) to computer via GPIB.

Step 2. Launch Visual Basic and create a new project.

Step 3. Import the following file to the project.

- age5260.bas or age5270.bas
  (e.g. \Program Files\VISA\winnt\include\age5270.bas)
- visa32.bas (e.g. \Program Files\VISA\winnt\include\visa32.bas)

Step 4. Open a form (e.g. Form1) in the project.

Step 5. Enter a program code as template. See Table 3-1 for example. The program code is written in Microsoft Visual Basic 6.0.

Step 6. Save the project as your template (e.g. \test\my_temp).

Table 3-1 Example Template Program Code for Visual Basic 6.0

```
Sub Main()                                                                    '1
'Starting the session ****************************************
Dim vi       As Long
Dim ret      As Long
Dim msg      As String
Dim err_msg  As String * 256
ret = age5270_init("GPIB::17::INSTR", VI_TRUE, VI_TRUE, vi)                   '7
If ((vi = VI_NULL) Or (ret < VI_SUCCESS)) Then
  msg = "Initialization failure." & Chr(10) & Chr(10) & "Status Code:  " & ret
  MsgBox msg, vbOKOnly, ""
  If (vi <> VI_NULL) Then
    ret = age5270_error_message(vi, ret, err_msg)
    msg = "Error:  " & ret & Chr(10) & Chr(10) & err_msg
    MsgBox msg, vbOKOnly, ""
  End If
  End
End If                                                                       '17

ret = age5270_reset(vi)                      'resets E5270B                  '19
ret = age5270_timeOut(vi, 60000)             'sets time out to 60 sec
ret = age5270_errorQueryDetect(vi, VI_TRUE)  'enables error detection
msg = "Click OK to start measurement."
MsgBox msg, vbOKOnly, ""                     'displays message box

perform_meas vi, ret           'calls perform_meas subprogram                 25
'ret = age5270_cmd(vi, "aa")   'sends an invalid command
'check_err vi, ret             'checks check_err subprogram operation
```

Line	Description
1	Beginning of the Main subprogram.
3 to 6	Declares variables used in this program.
7	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”.
8 to 17	Checks the status returned by the age5270_init function. If an error status is returned, displays a message box to show the error message, and stops the program execution.

```
Sub Main()                                                                    '1
'Starting the session ****************************************
Dim vi        As Long
Dim ret       As Long
Dim msg       As String
Dim err_msg   As String * 256
ret = age5270_init("GPIB::17::INSTR", VI_TRUE, VI_TRUE, vi)                   '7
If ((vi = VI_NULL) Or (ret < VI_SUCCESS)) Then
  msg = "Initialization failure." & Chr(10) & Chr(10) & "Status Code:  " & ret
  MsgBox msg, vbOKOnly, ""
  If (vi <> VI_NULL) Then
    ret = age5270_error_message(vi, ret, err_msg)
    msg = "Error:  " & ret & Chr(10) & Chr(10) & err_msg
    MsgBox msg, vbOKOnly, ""
  End If
  End
End If                                                                       '17

ret = age5270_reset(vi)                      'resets E5270B                  '19
ret = age5270_timeOut(vi, 60000)             'sets time out to 60 sec
ret = age5270_errorQueryDetect(vi, VI_TRUE)  'enables error detection
msg = "Click OK to start measurement."
MsgBox msg, vbOKOnly, ""                     'displays message box

perform_meas vi, ret           'calls perform_meas subprogram                 25
'ret = age5270_cmd(vi, "aa")   'sends an invalid command
'check_err vi, ret             'checks check_err subprogram operation
```

Line	Description
19 to 23	Resets the Agilent E5270B, sets the driver I/O time out to 60 seconds, and enables the automatic instrument error checking. Also opens a message box to confirm start of measurement.
25	Calls the perform_meas subprogram (line 38).
26 to 27	Should be deleted or commented out before executing the program. The lines are just used to check the operation of the check_err subprogram.

```
'Closing the session ****************************************
ret = age5270_close(vi)                                                    '30
check_err vi, ret
msg = "Click OK to stop the program."
MsgBox msg, vbOKOnly, ""

End Sub
'-------------------------------------------------------------------        36

Sub perform_meas(vi As Long, ret As Long)
   'insert program code
End Sub
'-------------------------------------------------------------------        41

Sub check_err(vi As Long, ret As Long)
 Dim inst_err    As Long
 Dim err_message As String * 250
 Dim msg         As String
 Dim retStatus   As Long
 If VI_SUCCESS > ret Then
  If (age5270_INSTR_ERROR_DETECTED = ret) Then
    retStatus = age5270_error_query(vi, inst_err, err_message)
    msg = "Instrument Error:  " & inst_err & Chr(10) & Chr(10) & err_message
    MsgBox msg, vbOKOnly, ""
  Else
    retStatus = age5270_error_message(vi, ret, err_message)
    msg = "Driver Error:  " & ret & Chr(10) & Chr(10) & err_message
    MsgBox msg, vbOKOnly, ""
  End If
 End If
End Sub
```

Line	Description
30	Disables the software connection with the Agilent E5270B.
31	Calls the check_err subprogram to check if an error status is returned for the line 30.
32 to 33	Opens a message box to confirm end of program.
35	End of the Main subprogram.
38 to 40	This is just the declaration of the perform_meas subprogram. Complete the subprogram that controls the E5270B, performs measurement, and displays/saves the results.
41 to last	Checks if the passed “ret” value indicates normal status, and returns to the line that called this subprogram. If the value indicates an instrument error status or a device error status, a message box will be displayed to show the error message.

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 Visual Basic.

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

Step 6. Open the form that contains the template code as shown in Table 3-1. On the code window, complete the perform_meas subprogram. Then use the Agilent E5260/E5270 VXI*plug&play* driver functions:

- age52*x*0_setSwitch to enable/disable the source/measurement channels
- age52*x*0_force, age52*x*0_setIv, etc. to set source outputs
- age52*x*0_spotMeas, age52*x*0_sweepIv, etc. to perform measurements
- age52*x*0_zeroOutput to disable source outputs

where, age52*x*0_ is age5260_ for the Agilent E5260 driver or age5270_ for the Agilent E5270 driver.

Step 7. Insert the code to display, store, or calculate data into the subprogram.

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

High Speed Spot Measurement

Table 3-2 explains example subprograms that enable/disable measurement channels (perform_meas), perform the high speed spot measurement (spot_meas), and display measurement result data (display_data). This example measures MOSFET drain current.

Table 3-2 High Speed Spot Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim pins(4) As Long  'SMU port numbers                       '3
pins(0) = 1          'SMU1: drain
pins(1) = 2          'SMU2: gate
pins(2) = 4          'SMU4: source
pins(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, pins(3), 1)                      '9
ret = age5270_setSwitch(vi, pins(2), 1)
ret = age5270_setSwitch(vi, pins(1), 1)
ret = age5270_setSwitch(vi, pins(0), 1)
check_err vi, ret                                            '13

spot_meas vi, ret, pins()                                    '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '17
check_err vi, ret

End Sub                                                      '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the spot_meas subprogram (next page) to perform spot measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub spot_meas(vi As Long, ret As Long, pins() As Long)                          '1

Dim vd As Double                                                                '3
Dim vg As Double
Dim idcomp As Double
Dim igcomp As Double
Dim meas As Double
Dim status As Long
vd = 0.5
idcomp = 0.05
vg = 0.5
igcomp = 0.01                                                                   '12

ret = age5270_force(vi, pins(3), age5270_VF_MODE, 0, 0, 0.05, 0)                '14
ret = age5270_force(vi, pins(2), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, pins(1), age5270_VF_MODE, 2, vg, igcomp, 0)
ret = age5270_force(vi, pins(0), age5270_VF_MODE, 2, vd, idcomp, 0)             '17
check_err vi, ret

ret = age5270_spotMeas(vi, pins(0), age5270_IM_MODE, 0, meas, status, 0)        '20
check_err vi, ret
ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                    '22
check_err vi, ret

display_data meas, status, vi, ret, pins()
End Sub
```

Line	Description
1	Beginning of the spot_meas subprogram.
3 to 12	Declares variables, and defines the value.
14 to 17	Applies voltage to device.
20	Performs the high speed spot measurement.
22	Sets the specified port to the zero output state.
18, 21, and 23	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
25	Calls the display_data subprogram (next page) to display measurement data.
26	End of the spot_meas subprogram.

```
Sub display_data(meas As Double, status As Long, vi As Long, ret
As Long, pins() As Long)

Dim title As String                                          '3
Dim value As String
Dim rbx As Integer
title = "Spot Measurement Result"                            '6

If status = 0 Then                                           '8
    value = "Id = " & meas * 1000 & " (mA)" & Chr(10) & Chr(10)
    value = value & "Do you want to perform measurement again?"
    rbx = MsgBox(value, vbYesNo + vbQuestion, title)
    If rbx = vbYes Then
        spot_meas vi, ret, pins()
    End If
Else
    value = "Status error.  Code = " & status
    MsgBox value, vbOKOnly, title
End If                                                       '18

End Sub
```

Line	Description
1	Beginning of the display_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 18	Displays measurement data on a message box if the measurement status is normal. If Yes is clicked on the message box, performs the spot_meas subprogram again. If No is clicked, returns to the perform_meas subprogram. Or displays error message on a message box if the status is abnormal.
20	End of the display_data subprogram.

Measurement Result Example

```
Id = 4.0565 (mA)

Do you want to perform measurement again?
```

Multi Channel Spot Measurement

Table 3-3 explains example subprograms that enable/disable measurement channels (perform_meas), perform the multi channel spot measurement (mspot_meas), and display measurement result data (display_data). This example measures bipolar transistor collector current and base current.

Table 3-3 Multi Channel Spot Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim pins(3) As Long  'SMU port numbers                       '3
pins(0) = 1          'SMU1: emitter
pins(1) = 2          'SMU2: base
pins(2) = 4          'SMU4: collector

ret = age5270_setSwitch(vi, pins(2), 1)                      '8
ret = age5270_setSwitch(vi, pins(1), 1)
ret = age5270_setSwitch(vi, pins(0), 1)
check_err vi, ret                                            '11

mspot_meas vi, ret, pins()                                   '13

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '15
check_err vi, ret

End Sub                                                      '18
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 6	Declares variables, and defines the value.
8 to 10	Enables measurement channels.
13	Calls the mspot_meas subprogram (next page) to perform multi channel spot measurement.
15	Disables measurement channels.
11 and 16	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
18	End of the perform_meas subprogram.

```
Sub mspot_meas(vi As Long, ret As Long, pins() As Long)       '1

Dim vc As Double                                              '3
Dim vb As Double
Dim ve As Double
Dim iccomp As Double
Dim ibcomp As Double
Dim iecomp As Double

ve =        0
iecomp =    0.2
vb =        0.7
ibcomp =    0.01
vc =        3
iccomp =    1                                                '15

Dim mch(3) As Long                                            '17
mch(0) =    pins(2)      'collector
mch(1) =    pins(1)      'base
mch(2) =    0

Dim mode(2) As Long
mode(0) =   1            'current measurement
mode(1) =   1            'current measurement

Dim range(2) As Double
range(0) = 0             'auto range
range(1) = 0             'auto range

Dim md(2) As Double
Dim st(2) As Long
Dim tm(2) As Double                                           '32
```

Line	Description
1	Beginning of the mspot_meas subprogram.
3 to 15	Declares variables used to set the source channels, and defines the value.
17 to 28	Declares variables used to set the measurement channels, and defines the value.
30 to 32	Declares variables used to keep the measurement data, status data, and time stamp data.

```
ret = age5270_resetTimestamp(vi)                                    '34
check_err vi, ret

ret = age5270_force(vi, pins(0), age5270_VF_MODE, 0, ve, iecomp,
0)
ret = age5270_force(vi, pins(1), age5270_VF_MODE, 0, vb, ibcomp,
0)
ret = age5270_force(vi, pins(2), age5270_VF_MODE, 0, vc, iccomp,
0)
check_err vi, ret                                                   '40

ret = age5270_measureM(vi, mch(0), mode(0), range(0), md(0),
st(0), tm(0))
check_err vi, ret                                                   '43

ret = age5270_zeroOutput(vi, age5270_CH_ALL)
check_err vi, ret                                                   '46

display_data md(), st(), tm(), vi, ret, pins()

End Sub
```

Line	Description
34	Resets time stamp.
37 to 39	Applies voltage to device.
42	Performs the multi channel spot measurement.
45	Sets the specified port to the zero output state.
35, 40, 43, and 46	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
48	Calls the display_data subprogram (next page) to display measurement data.
50	End of the mspot_meas subprogram.

```
Sub display_data(md() As Double, st() As Long, tm() As Double, vi As Long, ret As
Long, pins() As Long)                                                          '1

Dim title As String                                                            '3
Dim value As String
Dim rbx As Integer
title = "Spot Measurement Result"                                              '6

If st(0) = 0 Then                                                              '8
   value = "Ic = " & md(0) * 1000 & " (mA)"
   value = value & Chr(10) & "Time = " & tm(0) & "(sec)"
   If st(1) = 0 Then
      value = value & Chr(10) & Chr(10) & "Ib = " & md(1) * 1000 & " (mA)"
      value = value & Chr(10) & "Time = " & tm(1) & "(sec)"
      value = value & Chr(10) & Chr(10) & "hfe = " & md(0) / md(1)
      value = value & Chr(10) & Chr(10) & "Do you want to perform measurement
again?"
      rbx = MsgBox(value, vbYesNo + vbQuestion, title)
      If rbx = vbYes Then
         mspot_meas vi, ret, pins()
      End If
   Else
      value = "Base channel status error. Code = " & st(1)
      MsgBox value, vbOKOnly, title
   End If
Else
   value = "Collector channel status error.  Code = " & st(0)
   MsgBox value, vbOKOnly, title
End If                                                                         '27
End Sub
```

Line	Description
1	Beginning of the display_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 27	Displays measurement data on a message box if the measurement status is normal. If Yes is clicked on the message box, performs the mspot_meas subprogram again. If No is clicked, returns to the perform_meas subprogram. Or displays error message on a message box if the status is abnormal.
28	End of the display_data subprogram.

Measurement Result Example

```
Ic = 3.808 (mA)
Time = 0.061(sec)

Ib = 0.01883 (mA)
Time = 0.0636(sec)

hfe = 202.230483271375

Do you want to perform measurement again?
```

Pulsed Spot Measurement

Table 3-4 explains example subprograms that enable/disable measurement channels (perform_meas), perform the pulsed spot measurement (spot_meas), and display measurement result data (display_data). This example measures MOSFET drain current.

Table 3-4 Pulsed Spot Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                 '1

Dim pins(4) As Long  'SMU port numbers                    '3
pins(0) = 1          'SMU1: drain
pins(1) = 2          'SMU2: gate
pins(2) = 4          'SMU4: source
pins(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, pins(3), 1)                   '9
ret = age5270_setSwitch(vi, pins(2), 1)
ret = age5270_setSwitch(vi, pins(1), 1)
ret = age5270_setSwitch(vi, pins(0), 1)
check_err vi, ret                                         '13

spot_meas vi, ret, pins()                                 '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)            '17
check_err vi, ret

End Sub                                                   '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the spot_meas subprogram (next page) to perform pulsed spot measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub spot_meas(vi As Long, ret As Long, pins() As Long)        '1

Dim vd As Double                                              '3
Dim vg As Double
Dim idcomp As Double
Dim igcomp As Double
Dim meas As Double
Dim status As Long
vd = 0.5
idcomp = 0.05
vg = 0.5
igcomp = 0.01                                                 '12

Dim base As Double                                            '14
Dim width As Double
Dim period As Double
Dim hold As Double
base = 0
width = 0.001
period = 0.01
hold = 0.1                                                    '21

ret = age5270_setFilter(vi, pins(1), age5270_FLAG_OFF)        '23
ret = age5270_setPbias(vi, pins(1), age5270_VF_MODE, 2, base,
vg, width, period, hold, igcomp)
ret = age5270_force(vi, pins(3), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, pins(2), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, pins(0), age5270_VF_MODE, 2, vd, idcomp,
0)

check_err vi, ret                                             '29
```

Line	Description
1	Beginning of the spot_meas subprogram.
3 to 12	Declares variables for the dc sources, and defines the value.
14 to 21	Declares variables for the pulsed source, and defines the value.
23 to 24	Sets SMU filter off for the pulsed bias channel, and sets the pulsed bias source.
25 to 27	Applies voltage to device.
29	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.

```
ret = age5270_measureP(vi, pins(0), age5270_IM_MODE, 0, meas,
status, 0)                                                    '31
check_err vi, ret

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                  '34
check_err vi, ret

display_data meas, status, vi, ret, pins()                    '37

End Sub                                                       '39
```

Line	Description
31	Performs the pulsed spot measurement.
34	Sets the specified port to the zero output state.
32 and 35	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
37	Calls the display_data subprogram (next page) to display measurement data.
39	End of the spot_meas subprogram.

```
Sub display_data(meas As Double, status As Long, vi As Long, ret
As Long, pins() As Long)

Dim title As String                                          '3
Dim value As String
Dim rbx As Integer
title = "Spot Measurement Result"                            '6

If status = 0 Then                                           '8
    value = "Id = " & meas * 1000 & " (mA)" & Chr(10) & Chr(10)
    value = value & "Do you want to perform measurement again?"
    rbx = MsgBox(value, vbYesNo + vbQuestion, title)
    If rbx = vbYes Then
        spot_meas vi, ret, pins()
    End If
Else
    value = "Status error.  Code = " & status
    MsgBox value, vbOKOnly, title
End If                                                      '18

End Sub
```

Line	Description
1	Beginning of the display_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 18	Displays measurement data on a message box if the measurement status is normal. If Yes is clicked on the message box, performs the spot_meas subprogram again. If No is clicked, returns to the perform_meas subprogram. Or displays error message on a message box if the status is abnormal.
20	End of the display_data subprogram.

Measurement Result Example

```
Id = 4.075 (mA)

Do you want to perform measurement again?
```

Staircase Sweep Measurement

Table 3-5 explains example subprograms that enable/disable measurement channels (perform_meas), perform the staircase sweep measurement (sweep_meas), and save measurement result data into a file (save_data). This example measures MOSFET Id-Vd characteristics.

Table 3-5 **Staircase Sweep Measurement Example 1**

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim m(4) As Long  'SMU port numbers                          '3
m(0) = 1          'SMU1: drain
m(1) = 2          'SMU2: gate
m(2) = 4          'SMU4: source
m(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, m(3), 1)                         '9
ret = age5270_setSwitch(vi, m(2), 1)
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                            '13

sweep_meas vi, ret, m()                                      '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '17
check_err vi, ret

End Sub                                                      '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the sweep_meas subprogram (next page) to perform staircase sweep measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vd1          As Double                                    '3
Dim vd2          As Double
Dim idcomp       As Double
Dim vg1          As Double
Dim vg2          As Double
Dim igcomp       As Double
Dim hold         As Double
Dim delay        As Double
Dim s_delay      As Double
Dim p_comp       As Double
Dim nop1         As Long
Dim nop2         As Long
vd1 = 0
vd2 = 3
idcomp = 0.05
vg1 = 1
vg2 = 3
igcomp = 0.01
hold = 0
delay = 0
s_delay = 0
p_comp = 0
nop1 = 11
nop2 = 3
Dim i As Integer
Dim j As Integer
Dim n As Long
n = nop1 * nop2                                              '30

Dim msg              As String
Dim rep              As Long                                 '33
Dim sc()             As Double 'primary sweep output data
Dim md()             As Double 'sweep measurement data
Dim st()             As Long   'status data at each step
Dim tm()             As Double 'time data at each step
Dim dvg()            As Double 'secondary sweep output data
ReDim Preserve sc(n) As Double
ReDim Preserve md(n) As Double
ReDim Preserve st(n) As Long
ReDim Preserve tm(n) As Double
ReDim Preserve dvg(nop2) As Double                           '43
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 30	Declares variables, and defines the value.
33 to 43	Declares variables used to keep source data, measurement data, status data, and time stamp data. Also defines array size.

```
ret = age5270_resetTimestamp(vi)                                    '45
check_err vi, ret

ret = age5270_force(vi, m(3), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, 0, 0.05, 0)

Dim d_vg As Double 'secondary sweep step value (delta)          '51
If nop2 = 1 Then
    d_vg = 0
Else
    d_vg = (vg2 - vg1) / (nop2 - 1)
End If                                                          '56

Dim vg As Double   'secondary sweep source output               '58
vg = vg1

i = 0              'array counter for sweepIv returned data     '61
```

Line	Description
45	Resets time stamp.
46	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
48 to 49	Applies voltage to device.
51 to 56	Declares a variable, and defines the value. This variable is used for the step value of the secondary sweep source.
58 to 59	Declares a variable, and defines the value. This variable is used for the output value of the secondary sweep source.
61	Sets the array counter i to 0.

```
For j = 1 To nop2          'array counter for secondary sweep output data      '63
    dvg(j - 1) = vg
    ret = age5270_force(vi, m(1), age5270_VF_MODE, 0, vg, igcomp, 0)
    ret = age5270_setIv(vi, m(0), age5270_SWP_VF_SGLLIN, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
    check_err vi, ret
    ret = age5270_sweepIv(vi, m(0), age5270_IM_MODE, 0, rep, sc(i), md(i), st(i),
tm(i))
    check_err vi, ret
    vg = vg + d_vg
    If rep = nop1 Then
        i = i + nop1
    Else
        msg = rep & " measurement steps were returned. It must be " & nop1 & "
steps.   "
        MsgBox msg, vbOKOnly, ""
        ret = age5270_zeroOutput(vi, age5270_CH_ALL)
        check_err vi, ret
        GoTo Bottom_sub
    End If
Next j                                                                         '80

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                   '82
check_err vi, ret

save_data nop1, nop2, dvg(), md(), st(), sc(), tm(), vi, ret, m()              '85

Bottom_sub:
End Sub                                                                        '88
```

Line	Description
63 to 83	Measures MOSFET Id-Vd characteristics.
65 to 66	Applies voltage to device, and sets the voltage sweep source.
68	Performs the staircase sweep measurement.
71 to 80	Disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
85	Sets the specified port to the zero output state.
67, 69, 77, and 83	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
85	Calls the save_data subprogram (next page) to save measurement data.
88	End of the sweep_meas subprogram.

```
Sub save_data(nop1 As Long, nop2 As Long, dvg() As Double, md() As Double, st() As
Long, sc() As Double, tm() As Double, vi As Long, ret As Long, m() As Long)

Dim i     As Integer       'array counter for primary sweep                    '3
Dim j     As Integer       'array counter for secondary sweep
Dim val   As String        'data to be saved to a file
val = "Vg (V), Vd (V), Id (mA), Time (sec), Status"

For j = 1 To nop2                                                              '8
    For i = nop1 * (j - 1) To nop1 * j - 1
        val = val & Chr(13) & Chr(10) & dvg(j - 1) & "," & sc(i) & "," & md(i) *
1000 & "," & tm(i) & "," & st(i)
    Next i
Next j                                                                         '12

Dim fname   As String           'data file name                                '14
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data1.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum                                                                     '22
'displays data on a MsgBox
Dim title As String                                                            '24
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m() 'returns to sweep_meas if Yes is clicked.
End If                                                                         '32

End Sub
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 12	Creates data to be saved and displayed on a message box.
14 to 22	Saves measurement data into a file (C:\Agilent\data\data1.txt, CSV file).
24 to 32	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
34	End of the save_data subprogram.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1,0,-0.00011721,0.0703,0
1,0.3,3.1915,0.086,0
1,0.6,5.8795,0.0875,0
1,0.9,8.1215,0.0889,0
1,1.2,10.004,0.0904,0
1,1.5,11.64,0.0936,0
1,1.8,13.09,0.0948,0
1,2.1,14.385,0.0962,0
1,2.4,15.57,0.0972,0
1,2.7,16.63,0.0985,0
1,3,17.6,0.0995,0
2,0,-0.000117215,0.1983,0
2,0.3,4.178,0.2168,0
2,0.6,7.9075,0.2182,0
2,0.9,11.193,0.2197,0
2,1.2,14.035,0.2232,0
2,1.5,16.49,0.2242,0
2,1.8,18.59,0.2255,0
2,2.1,20.44,0.2265,0
2,2.4,22.095,0.2277,0
2,2.7,23.575,0.229,0
2,3,24.94,0.2301,0
3,0,0.00050875,0.3391,0
3,0.3,5.0385,0.3468,0
3,0.6,9.6655,0.3483,0
3,0.9,13.88,0.3517,0
3,1.2,17.65,0.353,0
3,1.5,21.005,0.354,0
3,1.8,23.935,0.3554,0
3,2.1,26.515,0.3564,0
3,2.4,28.775,0.3577,0
3,2.7,30.77,0.359,0
3,3,32.575,0.3601,0

Data save completed.

Do you want to perform measurement again?
```

Table 3-6 explains example subprograms that enable/disable measurement channels (perform_meas), perform the staircase sweep measurement (sweep_meas), and save measurement result data into a file (save_data). This example measures MOSFET Id-Vg characteristics. The subprogram uses the synchronous sweep source set by the age5270_setSweepSync function.

Table 3-6 Staircase Sweep Measurement Example 2

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim m(4) As Long  'SMU port numbers                          '3
m(0) = 1          'SMU1: drain
m(1) = 2          'SMU2: gate
m(2) = 4          'SMU4: source
m(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, m(3), 1)                         '9
ret = age5270_setSwitch(vi, m(2), 1)
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                            '13

sweep_meas vi, ret, m()                                      '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '17
check_err vi, ret

End Sub                                                      '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the sweep_meas subprogram (next page) to perform staircase sweep measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vpri1     As Double                                       '3
Dim vpri2     As Double
Dim vsyn1     As Double
Dim vsyn2     As Double
Dim vcon1     As Double
Dim vcon2     As Double
Dim i1comp    As Double
Dim i2comp    As Double
Dim hold      As Double
Dim delay     As Double
Dim s_delay   As Double
Dim p1comp    As Double
Dim p2comp    As Double
Dim nop       As Long
vpri1 = 0
vpri2 = 3
i1comp = 0.01
vsyn1 = 0
vsyn2 = 3
i2comp = 0.05
hold = 0
delay = 0
s_delay = 0
p1comp = 0
p2comp = 0
nop = 11                                                      '28

Dim rep               As Long                                 '30
Dim sc()              As Double 'primary sweep output data
Dim md()              As Double 'sweep measurement data
Dim st()              As Long   'status data at each step
Dim tm()              As Double 'time data at each step
ReDim Preserve sc(nop) As Double
ReDim Preserve md(nop) As Double
ReDim Preserve st(nop) As Long
ReDim Preserve tm(nop) As Double                              '38
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 28	Declares variables, and defines the value.
30 to 38	Declares variables used to keep source data, measurement data, status data, and time stamp data. Also defines array size.

```
ret = age5270_resetTimestamp(vi)                                                    '40
ret = age5270_force(vi, m(3), age5270_VF_MODE, 0, vcon1, 0.05, 0)
ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, vcon2, 0.05, 0)
ret = age5270_setIv(vi, m(1), age5270_SWP_VF_SGLLIN, 0, vpri1, vpri2, nop, hold,
delay, s_delay, i1comp, p1comp)
check_err vi, ret
ret = age5270_setSweepSync(vi, m(0), age5270_VF_MODE, 0, vsyn1, vsyn2, i2comp,
P2comp)
check_err vi, ret                                                                   '46

ret = age5270_sweepIv(vi, m(0), age5270_IM_MODE, 0, rep, sc(0), md(0), st(0),
tm(0))
check_err vi, ret

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                        '51

Dim msg As String
If rep = nop Then                                                                   '54
    save_data nop, md(), st(), sc(), tm(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be " & nop & " steps.   "
    MsgBox msg, vbOKOnly, ""
End If                                                                              '59

End Sub                                                                             '61
```

Line	Description
40	Resets time stamp.
41 to 42	Applies voltage to device.
43	Sets the primary sweep source.
45	Sets the synchronous sweep source.
48	Performs the staircase sweep measurement.
51	Sets the specified port to the zero output state.
44, 46, and 49	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
54 to 59	Calls the save_data subprogram to save measurement data. Or, displays a message box if the number of returned data is not equal to the nop value.
61	End of the sweep_meas subprogram.

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() As Double, tm() As
Double, vi As Long, ret As Long, m() As Long)                                  '1

Dim i     As Integer       'array counter for primary sweep                    '3
Dim val   As String        'data to be saved to a file
val = "Vg (V), Id (mA), Time (sec), Status"

For i = 0 To nop - 1                                                           '7
   val = val & Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & tm(i) & "," &
st(i)
Next i

Dim fname   As String           'data file name                                '11
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data2.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
'displays data on a MsgBox
Dim title As String                                                            '21
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m() 'returns to sweep_meas if Yes is clicked.
End If                                                                         '29

End Sub
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 5	Declares variables, and defines the value.
7 to 9	Creates data to be saved and displayed on a message box.
11 to 19	Saves measurement data into a file (C:\Agilent\data\data2.txt, CSV file).
21 to 29	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
31	End of the save_data subprogram.

Measurement Result Example

```
Vg (V), Id (mA), Time (sec), Status
0,-0.000098485,0.0714,0
0.3,2.338,0.0901,0
0.6,4.9295,0.0921,0
0.9,7.7645,0.0938,0
1.2,10.8095,0.0951,0
1.5,14.05,0.0985,0
1.8,17.465,0.1001,0
2.1,21.045,0.1016,0
2.4,24.755,0.1028,0
2.7,28.59,0.1043,0
3,32.54,0.1058,0

Data save completed.

Do you want to perform measurement again?
```

Table 3-7 explains example subprograms that enable/disable measurement channels (perform_meas), perform the staircase sweep measurement (sweep_meas), and save measurement result data into a file (save_data). This example uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 3-6, MOSFET Id-Vg measurement).

Table 3-7 Staircase Sweep Measurement Example 3

```
Sub perform_meas(vi As Long, ret As Long)                '1

Dim m(4) As Long  'SMU port numbers                      '3
m(0) = 1          'SMU1: drain
m(1) = 2          'SMU2: gate
m(2) = 4          'SMU4: source
m(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, m(3), 1)                     '9
ret = age5270_setSwitch(vi, m(2), 1)
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                        '13

sweep_meas vi, ret, m()                                  '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)           '17
check_err vi, ret

End Sub                                                  '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the sweep_meas subprogram (next page) to perform staircase sweep measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vpri1    As Double                                        '3
Dim vpri2    As Double
Dim vsyn1    As Double
Dim vsyn2    As Double
Dim vcon1    As Double
Dim vcon2    As Double
Dim i1comp   As Double
Dim i2comp   As Double
Dim hold     As Double
Dim delay    As Double
Dim s_delay  As Double
Dim p1comp   As Double
Dim p2comp   As Double
Dim nop      As Long
vpri1 = 0
vpri2 = 3
i1comp = 0.01
vsyn1 = 0
vsyn2 = 3
i2comp = 0.05
hold = 0
delay = 0
s_delay = 0
p1comp = 0
p2comp = 0
nop = 11                                                      '28

Dim rep              As Long                                  '30
Dim sc()             As Double 'primary sweep output data
Dim md()             As Double 'sweep measurement data
Dim st()             As Long   'status data at each step
Dim tm()             As Double 'time data at each step
ReDim Preserve sc(nop) As Double
ReDim Preserve md(nop) As Double
ReDim Preserve st(nop) As Long
ReDim Preserve tm(nop) As Double                              '38
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 28	Declares variables, and defines the value.
30 to 38	Declares variables used to keep source data, measurement data, status data, and time stamp data. Also defines array size.

```
ret = age5270_resetTimestamp(vi)                                                    '40
ret = age5270_force(vi, m(3), age5270_VF_MODE, 0, vcon1, 0.05, 0)
ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, vcon2, 0.05, 0)
ret = age5270_setIv(vi, m(1), age5270_SWP_VF_SGLLIN, 0, vpri1, vpri2, nop, hold,
delay, s_delay, i1comp, p1comp)
check_err vi, ret
ret = age5270_setNthSweep(vi, 2, m(0), age5270_VF_MODE, 0, vsyn1, vsyn2, i2comp,
P2comp)
check_err vi, ret                                                                   '46

ret = age5270_msweepIv(vi, m(0), age5270_IM_MODE, 0, rep, sc(0), md(0), st(0),
tm(0))
check_err vi, ret

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                        '51

Dim msg As String
If rep = nop Then                                                                   '54
    save_data nop, md(), st(), sc(), tm(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be " & nop & " steps."
    MsgBox msg, vbOKOnly, ""
End If                                                                              '59

End Sub                                                                             '61
```

Line	Description
40	Resets time stamp.
41 to 42	Applies voltage to device.
43	Sets the primary sweep source.
45	Sets the synchronous sweep source.
48	Performs the staircase sweep measurement.
51	Sets the specified port to the zero output state.
44, 46, and 49	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
54 to 59	Calls the save_data subprogram to save measurement data. Or, displays a message box if the number of returned data is not equal to the nop value.
61	End of the sweep_meas subprogram.

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() As Double, tm() As
Double, vi As Long, ret As Long, m() As Long)                                  '1

Dim i     As Integer       'array counter for primary sweep                    '3
Dim val   As String        'data to be saved to a file
val = "Vg (V), Id (mA), Time (sec), Status"

For i = 0 To nop - 1                                                           '7
   val = val & Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & tm(i) & "," &
st(i)
Next i

Dim fname   As String           'data file name                                '11
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data3.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
'displays data on a MsgBox
Dim title As String                                                            '21
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m() 'returns to sweep_meas if Yes is clicked.
End If                                                                         '29

End Sub
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 5	Declares variables, and defines the value.
7 to 9	Creates data to be saved and displayed on a message box.
11 to 19	Saves measurement data into a file (C:\Agilent\data\data3.txt, CSV file).
21 to 29	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
31	End of the save_data subprogram.

Measurement Result Example

```
Vg (V), Id (mA), Time (sec), Status
0,-0.000117215,0.0715,0
0.3,2.335,0.0904,0
0.6,4.928,0.092,0
0.9,7.767,0.0937,0
1.2,10.812,0.0953,0
1.5,14.045,0.0987,0
1.8,17.465,0.1,0
2.1,21.045,0.1015,0
2.4,24.765,0.103,0
2.7,28.6,0.1046,0
3,32.555,0.1058,0

Data save completed.

Do you want to perform measurement again?
```

Multi Channel Sweep Measurement

Table 3-8 explains example subprograms that enable/disable measurement channels (perform_meas), perform the multi channel sweep measurement (sweep_meas), and save measurement result data into a file (save_data). This example measures bipolar transistor Ic-Vb and Ib-Vb characteristics.

Table 3-8 Multi Channel Sweep Measurement Example 1

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim m(3) As Long  'SMU port numbers                          '3
m(0) = 2          'SMU1: base
m(1) = 4          'SMU2: collector
m(2) = 1          'SMU4: emitter

ret = age5270_setSwitch(vi, m(2), 1)                         '8
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                            '11

sweep_meas vi, ret, m()                                      '13

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '15
check_err vi, ret

End Sub                                                      '18
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 6	Declares variables, and defines the value.
8 to 10	Enables measurement channels.
13	Calls the sweep_meas subprogram (next page) to perform multi channel sweep measurement.
15	Disables measurement channels.
11 and 16	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
18	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vc        As Double                                       '3
Dim ve        As Double
Dim vb1       As Double
Dim vb2       As Double
Dim iccomp    As Double
Dim ibcomp    As Double
Dim iecomp    As Double
Dim hold      As Double
Dim delay     As Double
Dim s_delay   As Double
Dim pcomp     As Double
Dim nop       As Long
Dim n         As Long
Dim smpl      As Long
vc =          3
iccomp =      0.1
ve =          0
iecomp =      0.1
vb1 =         0.3
vb2 =         0.8
ibcomp =      0.001
hold =        0
delay =       0
s_delay =     0
pcomp =       0
nop =         11
smpl =        5
n =           nop *2                                          '30
Dim msg       As String
Dim mch(3)    As Long                                         '32
Dim mode(2)   As Long
Dim range(2)  As Double
mch(0) =      m(0)          'base
mch(1) =      m(1)          'collector
mch(2) =      0
mode(0) =     1             'current measurement
mode(1) =     1             'current measurement
range(0) =    -0.001        '  1 mA range fixed
range(1) =    -0.1          '100 mA range fixed               '41
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 30	Declares variables used to set the source channels, and defines the value.
32 to 41	Declares variables used to set the measurement channels, and defines the value.

```
Dim sc()               As Double  'primary sweep output data                  '43
Dim md()               As Double  'sweep measurement data
Dim st()               As Long    'status data at each step
Dim tm()               As Double  'time data at each step
ReDim Preserve sc(nop) As Double
ReDim Preserve md(n)   As Double
ReDim Preserve st(n)   As Long
ReDim Preserve tm(n)   As Double                                              '50

ret = age5270_setAdc(vi, age5270_HSPEED_ADC, age5270_INTEG_MANUAL, smpl,
age5270_FLAG_OFF)                                                             '52
ret = age5270_setAdcType(vi, age5270_CH_ALL, age5270_HSPEED_ADC)
ret = age5270_resetTimestamp(vi)
check_err vi, ret

ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, ve, iecomp, 0)              '57
ret = age5270_force(vi, m(1), age5270_VF_MODE, 0, vc, iccomp, 0)
ret = age5270_setIv(vi, m(0), age5270_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp)
check_err vi, ret

ret = age5270_sweepMiv(vi, mch(0), mode(0), range(0), rep, sc(0), md(0), st(0),
tm(0))                                                                        '62
check_err vi, ret

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                  '65
check_err vi, ret
```

Line	Description
43 to 50	Declares variables used to keep the measurement data, status data, and time stamp data. Also defines array size.
52 to 53	Sets the high speed ADC, and selects it for all measurement channels.
54	Resets time stamp.
57 to 59	Applies voltage to device, and sets the staircase sweep source.
62	Performs multi channel sweep measurement by the age5270_sweepMiv function.
65	Sets the specified port to the zero output state.
55, 60, 63, and 66	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.

```
If rep = nop Then                                                    '68
    save_data nop, md(), st(), sc(), tm(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be "
& nop & " steps.   "
    MsgBox msg, vbOKOnly, ""
End If                                                               '73

End Sub                                                              '75
```

Line	Description
68 to 73	Calls the save_data subprogram to save measurement data. Or, displays a message box if the number of returned data is not equal to the nop value.
75	End of the sweep_meas subprogram.

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() As
Double, tm() As Double, vi As Long, ret As Long, m() As Long)

Dim i     As Integer       'array counter for primary sweep  '3
Dim val   As String        'data to be saved to a file
val = "Vb (V), Ib (mA), Ic (mA), Time_b (sec), Time_c (sec),
Status_b, Status_c"

For i = 0 To nop - 1                                                 '7
val = val & Chr(13) & Chr(10) & sc(i) & "," & md(2 * i) * 1000 &
"," & md(2 * i + 1) * 1000
val = val & "," & tm(2 * i) & "," & tm(2 * i + 1) & "," & st(2 *
i) & "," & st(2 * i + 1)
Next i
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 5	Declares variables, and defines the value.
7 to 10	Creates data to be saved and displayed on a message box.

```
Dim fname   As String           'data file name             '12
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data4.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
'displays data on a MsgBox
Dim title As String                                          '22
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform
measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m()
End If                                                       '30

End Sub
```

Line	Description
12 to 20	Saves measurement data into a file (C:\Agilent\data\data4.txt, CSV file).
22 to 30	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
32	End of the save_data subprogram.

Measurement Result Example

```
Vb (V), Ib (mA), Ic (mA), Time_b (sec), Time_c (sec), Status_b,
Status_c
0.3,0,-0.005,0.0568,0.1427,0,0
0.35,0,-0.005,0.2288,0.3147,0,0
0.4,0,-0.005,0.4008,0.4867,0,0
0.45,0,-0.005,0.5728,0.6587,0,0
0.5,0,0,0.7448,0.8306,0,0
0.55,0.0001,0.015,0.9168,1.0027,0,0
0.6,0.0005,0.085,1.0888,1.1746,0,0
0.65,0.00305,0.605,1.2608,1.3467,0,0
0.7,0.01915,3.89,1.4328,1.5186,0,0
0.75,0.09975,19.625,1.6048,1.6906,0,0
0.8,0.34745,59.38,1.7768,1.8626,0,0

Data save completed.

Do you want to perform measurement again?
```

Table 3-9 explains example subprograms that enable/disable measurement channels (perform_meas), perform the multi channel sweep measurement (sweep_meas), and save measurement result data into a file (save_data). The following subprogram uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 3-8, bipolar transistor Ic-Vb and Ib-Vb measurement).

Table 3-9 Multi Channel Sweep Measurement Example 2

```
Sub perform_meas(vi As Long, ret As Long)                 '1

Dim m(3) As Long  'SMU port numbers                       '3
m(0) = 2          'SMU1: base
m(1) = 4          'SMU2: collector
m(2) = 1          'SMU4: emitter

ret = age5270_setSwitch(vi, m(2), 1)                      '8
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                         '11

sweep_meas vi, ret, m()                                   '13

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)            '15
check_err vi, ret

End Sub                                                   '18
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 6	Declares variables, and defines the value.
8 to 10	Enables measurement channels.
13	Calls the sweep_meas subprogram (next page) to perform multi channel sweep measurement.
15	Disables measurement channels.
11 and 16	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
18	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vc        As Double                                       '3
Dim ve        As Double
Dim vb1       As Double
Dim vb2       As Double
Dim iccomp    As Double
Dim ibcomp    As Double
Dim iecomp    As Double
Dim hold      As Double
Dim delay     As Double
Dim s_delay   As Double
Dim pcomp     As Double
Dim nop       As Long
Dim n         As Long
Dim smpl      As Long
vc =          3
iccomp =      0.1
ve =          0
iecomp =      0.1
vb1 =         0.3
vb2 =         0.8
ibcomp =      0.001
hold =        0
delay =       0
s_delay =     0
pcomp =       0
nop =         11
smpl =        5
n =           nop *2                                          '30
Dim msg       As String
Dim mch(3)    As Long                                         '32
Dim mode(2)   As Long
Dim range(2)  As Double
mch(0) =      m(0)          'base
mch(1) =      m(1)          'collector
mch(2) =      0
mode(0) =     1             'current measurement
mode(1) =     1             'current measurement
range(0) =    -0.001        '  1 mA range fixed
range(1) =    -0.1          '100 mA range fixed              '41
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 30	Declares variables used to set the source channels, and defines the value.
32 to 41	Declares variables used to set the measurement channels, and defines the value.

```
Dim sc()               As Double  'primary sweep output data                  '43
Dim md()               As Double  'sweep measurement data
Dim st()               As Long    'status data at each step
Dim tm()               As Double  'time data at each step
ReDim Preserve sc(nop) As Double
ReDim Preserve md(n)   As Double
ReDim Preserve st(n)   As Long
ReDim Preserve tm(n)   As Double                                              '50

ret = age5270_setAdc(vi, age5270_HSPEED_ADC, age5270_INTEG_MANUAL, smpl,
age5270_FLAG_OFF)                                                             '52
ret = age5270_setAdcType(vi, age5270_CH_ALL, age5270_HSPEED_ADC)
ret = age5270_resetTimestamp(vi)
check_err vi, ret

ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, ve, iecomp, 0)              '57
ret = age5270_force(vi, m(1), age5270_VF_MODE, 0, vc, iccomp, 0)
ret = age5270_setIv(vi, m(0), age5270_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp)
check_err vi, ret

ret = age5270_msweepMiv(vi, mch(0), mode(0), range(0), rep, sc(0), md(0), st(0),
tm(0))                                                                        '62
check_err vi, ret

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                  '65
check_err vi, ret
```

Line	Description
43 to 50	Declares variables used to keep the measurement data, status data, and time stamp data.
52 to 53	Sets the high speed ADC, and selects it for all measurement channels.
54	Resets time stamp.
57 to 59	Applies voltage to device, and sets the staircase sweep source.
62	Performs multi channel sweep measurement by the age5270_msweepMiv function.
65	Sets the specified port to the zero output state.
55, 60, 63, and 66	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.

```
If rep = nop Then                                                    '68
    save_data nop, md(), st(), sc(), tm(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be "
& nop & " steps.    "
    MsgBox msg, vbOKOnly, ""
End If                                                               '73

End Sub                                                              '75
```

Line	Description
68 to 73	Calls the save_data subprogram to save measurement data. Or, displays a message box if the number of returned data is not equal to the nop value.
75	End of the sweep_meas subprogram.

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() As
Double, tm() As Double, vi As Long, ret As Long, m() As Long)

Dim i     As Integer       'array counter for primary sweep  '3
Dim val   As String        'data to be saved to a file
val = "Vb (V), Ib (mA), Ic (mA), Time_b (sec), Time_c (sec),
Status_b, Status_c"

For i = 0 To nop - 1                                                 '7
val = val & Chr(13) & Chr(10) & sc(i) & "," & md(2 * i) * 1000 &
"," & md(2 * i + 1) * 1000
val = val & "," & tm(2 * i) & "," & tm(2 * i + 1) & "," & st(2 *
i) & "," & st(2 * i + 1)
Next i
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 5	Declares variables, and defines the value.
7 to 10	Creates data to be saved and displayed on a message box.

```
Dim fname   As String           'data file name             '12
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data5.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
'displays data on a MsgBox
Dim title As String                                          '22
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform
measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m()
End If                                                       '30

End Sub
```

Line	Description
12 to 20	Saves measurement data into a file (C:\Agilent\data\data5.txt, CSV file).
22 to 30	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
32	End of the save_data subprogram.

Measurement Result Example

```
Vb (V), Ib (mA), Ic (mA), Time_b (sec), Time_c (sec), Status_b,
Status_c
0.3,0,-0.005,0.057,0.057,0,0
0.35,0,-0.005,0.1434,0.1434,0,0
0.4,0,-0.005,0.23,0.23,0,0
0.45,0,-0.005,0.3164,0.3164,0,0
0.5,0,-0.005,0.403,0.403,0,0
0.55,0.0001,0.01,0.489,0.489,0,0
0.6,0.0005,0.085,0.5754,0.5754,0,0
0.65,0.00305,0.595,0.662,0.662,0,0
0.7,0.0191,3.855,0.7484,0.7484,0,0
0.75,0.0993,19.255,0.835,0.835,0,0
0.8,0.34475,57.825,0.9214,0.9214,0,0

Data save completed.

Do you want to perform measurement again?
```

Pulsed Sweep Measurement

Table 3-10 explains example subprograms that enable/disable measurement channels (perform_meas), perform the pulsed sweep measurement (sweep_meas), and save measurement result data into a file (save_data). This example measures bipolar transistor Ic-Vc characteristics.

Table 3-10 Pulsed Sweep Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim m(3) As Long  'SMU port numbers                          '3
m(0) = 4          'SMU4: collector
m(1) = 2          'SMU2: base
m(2) = 1          'SMU1: emitter

ret = age5270_setSwitch(vi, m(2), 1)                         '8
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                            '11

sweep_meas vi, ret, m()                                      '13

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '15
check_err vi, ret

End Sub                                                      '18
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 6	Declares variables, and defines the value.
8 to 10	Enables measurement channels.
13	Calls the sweep_meas subprogram (next page) to perform pulsed sweep measurement.
15	Disables measurement channels.
11 and 16	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
18	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vc1          As Double                                    '3
Dim vc2          As Double
Dim iccomp       As Double
Dim ib1          As Double
Dim ib2          As Double
Dim vbcomp       As Double
Dim hold         As Double
Dim width        As Double
Dim period       As Double
Dim base         As Double
Dim smpl         As Double
Dim nop1         As Long
Dim nop2         As Long
vc1 = 0
vc2 = 3
iccomp = 0.05
ib1 = 0.00005    ' 50 uA
ib2 = 0.00015    '150 uA
vbcomp = 5
hold = 0.1
width = 0.001
period = 0.01
base = 0
smpl = 5
nop1 = 11
nop2 = 3

Dim i As Integer
Dim j As Integer
Dim n As Long
n = nop1 * nop2                                              '33
Dim msg              As String
Dim rep              As Long                                 '35
Dim sc()             As Double 'primary sweep output data
Dim md()             As Double 'sweep measurement data
Dim st()             As Long   'status data at each step
Dim tm()             As Double 'time data at each step
Dim dib()            As Double 'secondary sweep output data
ReDim Preserve sc(n) As Double
ReDim Preserve md(n) As Double
ReDim Preserve st(n) As Long
ReDim Preserve tm(n) As Double
ReDim Preserve dib(nop2) As Double                           '45
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 33	Declares variables, and defines the value.
35 to 45	Declares variables used to keep source data, measurement data, status data, and time stamp data. Also defines array size.

```
ret = age5270_setAdc(vi, age5270_HSPEED_ADC,
age5270_INTEG_MANUAL, smpl, age5270_FLAG_OFF)                    '47
ret = age5270_setAdcType(vi, age5270_CH_ALL,
age5270_HSPEED_ADC)
ret = age5270_resetTimestamp(vi)
check_err vi, ret                                                '50

ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, 0, 0.05, 0)

Dim d_ib As Double   'secondary sweep step value (delta)        '54
If nop2 = 1 Then
    d_ib = 0
Else
    d_ib = (ib2 - ib1) / (nop2 - 1)
End If                                                           '59

Dim ibo As Double    'secondary sweep source output             '61
ibo = ib1

i = 0                'array counter for sweepIv returned data
```

Line	Description
47 to 48	Sets the high speed ADC, and selects it for all measurement channels.
49	Resets time stamp.
50	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
52	Applies voltage to device.
54 to 59	Declares a variable, and defines the value. This variable is used for the step value of the secondary sweep source.
61 to 62	Declares a variable, and defines the value. This variable is used for the output value of the secondary sweep source.
64	Sets the array counter i to 0.

```
For j = 1 To nop2                                                          '66
  dib(j - 1) = ibo
  ret = age5270_force(vi, m(1), age5270_IF_MODE, 0, ibo, vbcomp, 0)
  ret = age5270_setPiv(vi, m(0), age5270_SWP_VF_SGLLIN, 0, base, vc1, vc2, nop1,
hold, width, period, iccomp)
  check_err vi, ret
  ret = age5270_sweepPiv(vi, m(0), age5270_IM_MODE, 0, rep, sc(i), md(i), st(i),
tm(i))
  check_err vi, ret
  ibo = ibo + d_ib
  If rep = nop1 Then                                                       '74
     i = i + nop1
  Else
     msg = rep & " measurement steps were returned. It must be " & nop1 & " steps."
     MsgBox msg, vbOKOnly, ""
     ret = age5270_zeroOutput(vi, age5270_CH_ALL)
     check_err vi, ret
     GoTo Bottom_sub
  End If                                                                   '82
Next j                                                                     '83

ret = age5270_zeroOutput(vi, age5270_CH_ALL)                               '85
check_err vi, ret

save_data nop1, nop2, md(), st(), sc(), tm(), dib(), vi, ret, m()          '88

Bottom_sub:
End Sub                                                                    '91
```

Line	Description
68 to 69	Applies current to device, and sets the pulsed voltage sweep source.
71	Performs the pulsed sweep measurement.
73 to 82	Disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
85	Sets the specified port to the zero output state.
70, 72, 80, and 86	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
88	Calls the save_data subprogram (next page) to save measurement data.
91	End of the sweep_meas subprogram.

```
Sub save_data(nop1 As Long, nop2 As Long, md() As Double, st() As Long, sc() As
Double, tm() As Double, dib() As Double, vi As Long, ret As Long, m() As Long)  '1

Dim i   As Integer              'array counter for sweepPiv returned data       '3
Dim j   As Integer              'array counter for secondary sweep output data
Dim val As String               'data to be saved to a file

val = "Ib (uA), Vc (V), Ic (mA), Time (sec), Status"

For j = 1 To nop2                                                               '9
  For i = nop1 * (j - 1) To nop1 * j - 1
     val = val & Chr(13) & Chr(10) & dib(j - 1) * 1000000# & "," & sc(i) & "," &
md(i) * 1000 & "," & tm(i) & "," & st(i)
  Next i
Next j                                                                          '14

Dim fname   As String           'data file name                                 '16
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data6.txt"
fnum = 1
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
Dim title As String                                                             '23
Dim rbx As Integer
title = "Pulsed Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
   sweep_meas vi, ret, m()       'returns to sweep_meas if Yes is clicked.
End If

End Sub                                                                         '33
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 7	Declares variables, and defines the value.
9 to 14	Creates data to be saved and displayed on a message box.
16 to 22	Saves measurement data into a file (C:\Agilent\data\data6.txtt, CSV file).
23 to 31	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
33	End of the save_data subprogram.

Measurement Result Example

```
Ib (uA), Vc (V), Ic (mA), Time (sec), Status
50,0,-0.05,0.1539,0
50,0.3,8.965,0.1639,0
50,0.6,9.705,0.1739,0
50,0.9,9.735,0.1839,0
50,1.2,9.765,0.1939,0
50,1.5,9.805,0.2039,0
50,1.8,9.83,0.2139,0
50,2.1,9.835,0.2239,0
50,2.4,9.85,0.2339,0
50,2.7,9.9,0.2439,0
50,3,9.915,0.2539,0
100,0,-0.1,0.4039,0
100,0.3,15.725,0.4139,0
100,0.6,18.115,0.4239,0
100,0.9,18.715,0.4339,0
100,1.2,18.84,0.4439,0
100,1.5,18.925,0.4539,0
100,1.8,19.015,0.4639,0
100,2.1,19.045,0.4739,0
100,2.4,19.12,0.4839,0
100,2.7,19.175,0.4939,0
100,3,19.215,0.5039,0
150,0,-0.15,0.6539,0
150,0.3,21.065,0.6639,0
150,0.6,24.54,0.6739,0
150,0.9,26.47,0.6839,0
150,1.2,27.19,0.6939,0
150,1.5,27.405,0.7039,0
150,1.8,27.605,0.7139,0
150,2.1,27.71,0.7239,0
150,2.4,27.795,0.7339,0
150,2.7,27.885,0.7439,0
150,3,27.955,0.7539,0

Data save completed.

Do you want to perform measurement again?
```

Staircase Sweep with Pulsed Bias Measurement

Table 3-11 explains example subprograms that enable/disable measurement channels (perform_meas), perform the staircase sweep with pulsed bias measurement (sweep_meas), and save measurement result data into a file (save_data). This example measures MOSFET Id-Vd characteristics.

Table 3-11 Staircase Sweep with Pulsed Bias Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim m(4) As Long  'SMU port numbers                          '3
m(0) = 1          'SMU1: drain
m(1) = 2          'SMU2: gate
m(2) = 4          'SMU4: source
m(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, m(3), 1)                         '9
ret = age5270_setSwitch(vi, m(2), 1)
ret = age5270_setSwitch(vi, m(1), 1)
ret = age5270_setSwitch(vi, m(0), 1)
check_err vi, ret                                            '13

sweep_meas vi, ret, m()                                      '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '17
check_err vi, ret

End Sub                                                      '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the sweep_meas subprogram (next page) to perform staircase sweep measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub sweep_meas(vi As Long, ret As Long, m() As Long)          '1

Dim vd1          As Double                                    '3
Dim vd2          As Double
Dim idcomp       As Double
Dim vg1          As Double
Dim vg2          As Double
Dim igcomp       As Double
Dim hold         As Double
Dim delay        As Double
Dim s_delay      As Double
Dim p_comp       As Double

vd1 = 0
vd2 = 3
idcomp = 0.05
vg1 = 1
vg2 = 3
igcomp = 0.01
hold = 0
delay = 0
s_delay = 0
p_comp = 0

Dim nop1         As Long
Dim nop2         As Long
nop1 = 11
nop2 = 3

Dim i As Integer
Dim j As Integer
Dim n As Long
n = nop1 * nop2                                              '33
Dim msg              As String
Dim rep              As Long                                 '35
Dim sc()             As Double 'primary sweep output data
Dim md()             As Double 'sweep measurement data
Dim st()             As Long   'status data at each step
Dim tm()             As Double 'time data at each step
Dim dvg()            As Double 'secondary sweep output data
ReDim Preserve sc(n) As Double
ReDim Preserve md(n) As Double
ReDim Preserve st(n) As Long
ReDim Preserve tm(n) As Double
ReDim Preserve dvg(nop2) As Double                           '45
```

Line	Description
1	Beginning of the sweep_meas subprogram.
3 to 33	Declares variables for source channels, and defines the value.
35 to 45	Declares variables used to keep source data, measurement data, status data, and time stamp data. Also defines array size.

```
ret = age5270_resetTimestamp(vi)                                    '47
check_err vi, ret

ret = age5270_force(vi, m(3), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, m(2), age5270_VF_MODE, 0, 0, 0.05, 0)

Dim d_vg As Double 'secondary sweep step value (delta)              '53
If nop2 = 1 Then
    d_vg = 0
Else
    d_vg = (vg2 - vg1) / (nop2 - 1)
End If                                                              '58

Dim vg As Double   'secondary sweep source output                   '60
vg = vg1

i = 0              'array counter for sweepIv returned data         '63

Dim width       As Double                                           '65
Dim period      As Double
Dim p_hold      As Double
width = 0.001
period = 0.01
p_hold = 0.1                                                        '70
```

Line	Description
47	Resets time stamp.
48	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
50 to 51	Applies voltage to device.
53 to 58	Declares a variable, and defines the value. This variable is used for the step value of the secondary sweep source.
60 to 61	Declares a variable, and defines the value. This variable is used for the output value of the secondary sweep source.
63	Sets the array counter i to 0.
65 to 70	Declares variables for the pulsed source, and defines the value.

```
For j = 1 To nop2          'array counter for secondary sweep output data      '72
    dvg(j - 1) = vg
    ret = age5270_setPbias(vi, m(1), age5270_VF_MODE, 0, 0, vg, width, period,
p_hold, igcomp)
    ret = age5270_setIv(vi, m(0), age5270_SWP_VF_SGLLIN, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
    check_err vi, ret
    ret = age5270_sweepPbias(vi, m(0), age5270_IM_MODE, 0, rep, sc(i), md(i),
st(i), tm(i))
    check_err vi, ret
    vg = vg + d_vg
    If rep = nop1 Then
        i = i + nop1
    Else                                                                       '82
        msg = rep & " measurement steps were returned. It must be " & nop1 & "
steps."
        MsgBox msg, vbOKOnly, ""
        ret = age5270_zeroOutput(vi, age5270_CH_ALL)
        check_err vi, ret
        GoTo Bottom_sub
    End If                                                                     '88
Next j                                                                         '89
ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                   '90
check_err vi, ret

save_data nop1, nop2, dvg(), md(), st(), sc(), tm(), vi, ret, m()              '93

Bottom_sub:
End Sub
```

Line	Description
72 to 91	Measures MOSFET Id-Vd characteristics.
74 to 75	Sets the pulsed source and the voltage sweep source.
57	Performs the staircase sweep with pulsed bias measurement.
82 to 88	Disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
90	Sets the specified port to the zero output state.
76, 78, 86, and 91	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
93	Calls the save_data subprogram (next page) to save measurement data.
96	End of the sweep_meas subprogram.

```
Sub save_data(nop1 As Long, nop2 As Long, dvg() As Double, md() As Double, st() As
Long, sc() As Double, tm() As Double, vi As Long, ret As Long, m() As Long)

Dim i     As Integer       'array counter for primary sweep                    '3
Dim j     As Integer       'array counter for secondary sweep
Dim val   As String        'data to be saved to a file
val = "Vg (V), Vd (V), Id (mA), Time (sec), Status"

For j = 1 To nop2                                                              '8
    For i = nop1 * (j - 1) To nop1 * j - 1
        val = val & Chr(13) & Chr(10) & dvg(j - 1) & "," & sc(i) & "," & md(i) *
1000 & "," & tm(i) & "," & st(i)
    Next i
Next j                                                                         '12

Dim fname   As String           'data file name                                '14
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data7.txt"
fnum = 1

'saves data into the file specified by fname
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum                                                                     '22
'displays data on a MsgBox
Dim title As String                                                            '24
Dim rbx As Integer
title = "Sweep Measurement Result"
val = val & Chr(10) & Chr(10) & "Data save completed."
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
    sweep_meas vi, ret, m() 'returns to sweep_meas if Yes is clicked.
End If                                                                         '32

End Sub
```

Line	Description
1	Beginning of the save_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 12	Creates data to be saved and displayed on a message box.
14 to 22	Saves measurement data into a file (C:\Agilent\data\data7.txt, CSV file).
24 to 32	Displays measurement data on a message box. If Yes is clicked on the message box, performs the sweep_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
34	End of the save_data subprogram.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1,0,0,0.1664,0
1,0.3,3.205,0.1764,0
1,0.6,5.9,0.1864,0
1,0.9,8.15,0.1964,0
1,1.2,10.035,0.2064,0
1,1.5,11.68,0.2164,0
1,1.8,13.13,0.2264,0
1,2.1,14.425,0.2364,0
1,2.4,15.61,0.2464,0
1,2.7,16.675,0.2564,0
1,3,17.65,0.2664,0
2,0,-0.005,0.4182,0
2,0.3,4.205,0.4282,0
2,0.6,7.955,0.4382,0
2,0.9,11.245,0.4482,0
2,1.2,14.11,0.4582,0
2,1.5,16.55,0.4682,0
2,1.8,18.67,0.4782,0
2,2.1,20.52,0.4882,0
2,2.4,22.185,0.4982,0
2,2.7,23.67,0.5082,0
2,3,25.02,0.5182,0
3,0,0,0.6708,0
3,0.3,5.07,0.6808,0
3,0.6,9.73,0.6908,0
3,0.9,13.965,0.7008,0
3,1.2,17.76,0.7108,0
3,1.5,21.115,0.7208,0
3,1.8,24.07,0.7308,0
3,2.1,26.64,0.7408,0
3,2.4,28.91,0.7508,0
3,2.7,30.925,0.7608,0
3,3,32.71,0.7708,0

Data save completed.

Do you want to perform measurement again?
```

Breakdown Voltage Measurement

Table 3-12 explains example subprograms that enable/disable measurement channels (perform_meas), perform the quasi pulsed spot measurement (vbd_meas), and display measurement result data (display_data). This example measures bipolar transistor breakdown voltage.

Table 3-12 Breakdown Voltage Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                    '1

Dim pins(3) As Long  'SMU port numbers                       '3
pins(0) = 1          'SMU1: emitter
'pins(1) = 2          'SMU2: base - open
pins(2) = 4          'SMU4: collector

ret = age5270_setSwitch(vi, pins(2), 1)                      '8
'ret = age5270_setSwitch(vi, pins(1), 1)
ret = age5270_setSwitch(vi, pins(0), 1)
check_err vi, ret                                            '11

vbd_meas vi, ret, pins()                                     '13

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)               '15
check_err vi, ret

End Sub                                                      '18
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 6	Declares variables, and defines the value.
8 to 10	Enables measurement channels.
13	Calls the vbd_meas subprogram (next page) to perform breakdown voltage measurement.
15	Disables measurement channels.
11 and 16	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
18	End of the perform_meas subprogram.

```
Sub vbd_meas(vi As Long, ret As Long, pins() As Long)                          '1
Dim vstart As Double
Dim vstop  As Double
Dim vb     As Double
Dim ve     As Double
Dim iccomp As Double
Dim ibcomp As Double
Dim iecomp As Double
Dim hold As Double
Dim delay As Double
vstart =   0
vstop =    100         'interlock cable must be connected.
vb =       0.7
ve =       0
iccomp =   0.005
ibcomp =   0.01
iecomp =   0.1
hold =     0
delay =    0                                                                   '19

Dim meas   As Double                                                           '21
Dim status As Long

ret = age5270_force(vi, pins(0), age5270_VF_MODE, 0, ve, iecomp, 0)            '24
'ret = age5270_force(vi, pins(1), age5270_VF_MODE, 0, vb, ibcomp, 0)
ret = age5270_setBdv(vi, pins(2), 0, vstart, vstop, iccomp, hold, delay)
check_err vi, ret
ret = age5270_measureBdv(vi, age5270_SHORT_INTERVAL, meas, status)             '28
check_err vi, ret
ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                   '30
check_err vi, ret
display_data meas, status, vi, ret, pins()
End Sub
```

Line	Description
1	Beginning of the vbd_meas subprogram.
3 to 19	Declares variables for source channels, and defines the value.
21 to 22	Declares variables for the measurement data and the status data.
24 to 26	Applies voltage to device, and sets the quasi pulsed voltage source.
28	Performs the quasi pulsed spot measurement (breakdown voltage measurement).
30	Sets the specified port to the zero output state.
27, 29, and 31	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
32	Calls the display_data subprogram (shown below) to display measurement data.
33	End of the vbd_meas subprogram.

```
Sub display_data(meas As Double, status As Long, vi As Long, ret As Long, pins() As
Long)                                                                          '1
Dim title As String
Dim value As String
Dim rbx As Integer
title = "Vbd Measurement Result"
If status = 8 Then    'status=8 is returned when Vbd was measured normally     '6
   value = "Vbd = " & meas & " (V)"
Else
   value = "Vbd = " & meas & " (V)"
   value = value & Chr(10) & Chr(10) & "Status value = " & status
End If
value = value & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
rbx = MsgBox(value, vbYesNo + vbQuestion, title)
If rbx = vbYes Then
   vbd_meas vi, ret, pins()
End If                                                                         '16
End Sub
```

Line	Description
1	Beginning of the display_data subprogram.
2 to 5	Declares variables, and defines the value.
6 to 16	Displays measurement data on a message box if the measurement status is normal (8). Or displays error message on a message box if the status is abnormal. If Yes is clicked on the message box, performs the vbd_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.
17	End of the display_data subprogram.

Measurement Result Example

```
Vbd = 55.885 (V)

Do you want to perform measurement again?
```

Leakage Current Measurement

Table 3-13 explains example subprograms that enable/disable measurement channels (perform_meas), perform the quasi pulsed spot measurement (spot_meas), and display measurement result data (display_data). This example measures MOSFET drain current.

Table 3-13 Leakage Current Measurement Example

```
Sub perform_meas(vi As Long, ret As Long)                '1

Dim pins(4) As Long  'SMU port numbers                   '3
pins(0) = 1          'SMU1: drain
pins(1) = 2          'SMU2: gate
pins(2) = 4          'SMU4: source
pins(3) = 6          'SMU6: substrate

ret = age5270_setSwitch(vi, pins(3), 1)                  '9
ret = age5270_setSwitch(vi, pins(2), 1)
ret = age5270_setSwitch(vi, pins(1), 1)
ret = age5270_setSwitch(vi, pins(0), 1)
check_err vi, ret                                        '13

spot_meas vi, ret, pins()                                '15

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0)           '17
check_err vi, ret

End Sub                                                  '20
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 7	Declares variables, and defines the value.
9 to 12	Enables measurement channels.
15	Calls the spot_meas subprogram (next page) to perform spot measurement.
17	Disables measurement channels.
13 and 18	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
20	End of the perform_meas subprogram.

```
Sub spot_meas(vi As Long, ret As Long, pins() As Long)                          '1
Dim vstart As Double
Dim vstop  As Double
Dim vg     As Double
Dim idcomp As Double
Dim igcomp As Double
Dim hold   As Double
Dim delay  As Double
Dim meas   As Double
Dim status As Long
vstart =   -5
vstop =    5
idcomp =   0.05
vg =       0
igcomp =   0.01
hold =     0.1
delay =    0.001                                                                '17

ret = age5270_force(vi, pins(3), age5270_VF_MODE, 0, 0, 0.05, 0)                '19
ret = age5270_force(vi, pins(2), age5270_VF_MODE, 0, 0, 0.05, 0)
ret = age5270_force(vi, pins(1), age5270_VF_MODE, 2, vg, igcomp, 0)
ret = age5270_setIleak(vi, pins(0), 0, vstop, idcomp, vstart, hold, delay)      '22
check_err vi, ret
ret = age5270_measureIleak(vi, pins(0), age5270_SHORT_INTERVAL, meas, status)   '24
check_err vi, ret
ret = age5270_zeroOutput(vi, age5270_CH_ALL)                                    '26
check_err vi, ret
display_data meas, status, vi, ret, pins()                                      '28
End Sub
```

Line	Description
1	Beginning of the spot_meas subprogram.
3 to 17	Declares variables, and defines the value.
19 to 21	Applies voltage to device.
22	Sets the quasi pulsed voltage source.
24	Performs the quasi pulsed spot measurement (leakage current measurement).
26	Sets the specified port to the zero output state.
23, 25, and 27	Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line.
28	Calls the display_data subprogram (next page) to display measurement data.
29	End of the spot_meas subprogram.

```
Sub display_data(meas As Double, status As Long, vi As Long, ret
As Long, pins() As Long)

Dim title As String                                          '3
Dim value As String
Dim rbx As Integer
title = "Ileak Measurement Result"                           '6

If status = 0 Then                                           '8
    value = "Id = " & meas * 1000 & " (mA)" & Chr(10) & Chr(10)
    value = value & "Do you want to perform measurement again?"
    rbx = MsgBox(value, vbYesNo + vbQuestion, title)
    If rbx = vbYes Then
        spot_meas vi, ret, pins()
    End If
Else
    value = "Status error.  Code = " & status
    MsgBox value, vbOKOnly, title
End If                                                       '18

End Sub
```

Line	Description
1	Beginning of the display_data subprogram.
3 to 6	Declares variables, and defines the value.
8 to 18	Displays measurement data on a message box if the measurement status is normal. If Yes is clicked on the message box, performs the spot_meas subprogram again. If No is clicked, returns to the perform_meas subprogram. Or displays error message on a message box if the status is abnormal.
20	End of the display_data subprogram.

Measurement Result Example

```
Id = 12.775 (mA)

Do you want to perform measurement again?
```

4 Programming Examples for Visual Basic .NET Users

This chapter provides programming examples to perform the following measurements using the Agilent E5260/E5270 and the E5260/E5270 VXI*plug&play* driver.

- “Programming Basics”
- “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”

NOTE

About Program Code

Programming examples are provided as a subprogram that can be run with the project template shown in Table 4-1. To execute the program, insert the subprogram instead of the perform_meas subprogram in the template.

NOTE

Driver function name and instrument handle vi

Function name is different from the original name. And you do not need to specify the instrument handle vi for a function parameter. For example, age5270_reset(vi) must be entered as Mye5270.Reset() if you declare your E5270B as Mye5270.

For the available functions, see the function input aid of the driver. It will appear when you type the declared instrument name and a period (e.g. `Mye5270._`) on the code window. Also see the parameter input aid to know the parameters needed for the function. It will appear when you additionally type the function name and front-parenthesis (e.g. `Mye5270.Force(_ )`.

NOTE

To Start Program

If you create the measurement program by modifying the example code shown in Table 4-1, the program can be run by clicking the Run button on the Visual Basic main window. After that, a message box will appear. Then click OK to continue.

NOTE

For the Agilent E5260 Users

The example program code uses the Agilent E5270 VXI*plug&play* driver. So the modification is required for the Agilent E5260.

- Delete Mye5270.Asu, Mye5270.AsuLed, and Mye5270.SetAdcType. There is no replaceable function for them.
- Change the prefix of the function name to the *name* (e.g. Mye5260) as you declared for your Agilent E5260.
- Correct the parameter values for some functions. Available values are different for each SMU. See “Parameters” on page 2-7.
- Correct the syntax of the *name*.SetAdc function (e.g. Mye5260.SetAdc). See “age5260_setAdc” on page 2-38.

where, *name* means the declared instrument name (e.g. Mye5260).

Programming Basics

This section provides the basic information for programming using the Agilent E5260/E5270 VXI*plug&play* driver.

- "To Create Your Project Template"
- "To Create Measurement Program"

To Create Your Project Template

This section explains how to create a project template by using Microsoft Visual Basic .NET. Before starting programming, create your project template, and keep it as your reference. It will remove the conventional task in the future programming.

Step 1. Connect instrument (e.g. Agilent E5270) to computer via GPIB.

Step 2. Launch Visual Basic .NET and create a new project. The project type must be Agilent T&M Toolkit Projects.

Follow the Agilent T&M Toolkit New Project Wizard to create the project. Then select the following libraries to be imported additionally.

- Agilent.TMFramework.InstrumentDriverInterop
- Agilent.TMFramework.InstrumentDriverInterop.Design

Step 3. Click T&M Toolkit > Instrument Explorer to open Agilent Instrument Explorer. On the explorer, click Find Instrument icon to detect the instrument automatically.

Step 4. Click T&M Toolkit > Driver Wrapper Wizard to open Agilent Driver Wrapper Wizard. And follow the wizard to enable your desired VXI*plug&play* driver (e.g. AGE5270 - VXIplug&play).

Step 5. Right-click on the instrument icon (e.g. AGE5270 (:17) icon) in the Agilent Instrument Explorer, and click on Add Instrument Session to open Agilent Instrument Session Wizard. And follow the wizard to add the VXI*plug&play* session for the instrument (e.g. AGE5270).

Step 6. Open a module (e.g. Module1.vb) in the project.

Step 7. Enter a program code as template. See Table 4-1 for example.

Step 8. Save the project as your template (e.g. \test\my_temp).

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 Visual Basic .NET.

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

Step 6. Open the module that contains the template code as shown in Table 4-1. On the code window, complete the perform_meas subprogram. Then use the Agilent E5260/E5270 VXI*plug&play* driver functions:

- *name*.SetSwitch to enable/disable the source/measurement channels
- *name*.Force, SetIv, etc. to set source outputs
- *name*.SpotMeas, SweepIv, etc. to perform measurements
- *name*.ZeroOutput to disable source outputs

where, *name* must be the declared instrument name (e.g. Mye5270).

Step 7. Insert the code to display, store, or calculate data into the subprogram.

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

Table 4-1 Example Template Program Code for Visual Basic .NET

```
Imports Agilent.TMFramework                                                    '1
Imports Agilent.TMFramework.DataAnalysis
Imports Agilent.TMFramework.DataVisualization
Imports Agilent.TMFramework.InstrumentIO
Imports Agilent.TMFramework.InstrumentDriverInterop
Imports Agilent.TMFramework.InstrumentDriverInterop.Design
Imports Agilent.TMFramework.InstrumentDriverInterop.VxipnpWrappers

Module Module1

  Sub Main()                                                                   '11
    Dim Mye5270 As Age5270 = New Age5270("GPIB0::17::INSTR", True, True)
    Mye5270.Reset()
    Mye5270.TimeOut(60000)

    MsgBox("Click OK to start measurement.", vbOKOnly, "")
    Console.WriteLine("Measurement in progress. . ." & Chr(10))
    perform_meas(Mye5270)

    Mye5270.SetSwitch(0, 0)
    Mye5270.Close()
    MsgBox("Click OK to stop the program.", vbOKOnly, "")
    Console.WriteLine("Measurement completed." & Chr(10))
  End Sub                                                                      '24
```

Line	Description
1 to 7	These lines are necessary to use the Agilent E5260/E5270 VXI*plug&play* driver.
11 to 24	Main subprogram establishes the software connection with the Agilent E5270B, resets the E5270B, sets the driver I/O time out to 60 seconds, opens a message box to confirm the start of measurement, and pauses program execution until OK is clicked on the message box. By clicking OK, the program displays a message on the console window and calls the perform_meas subprogram that is used to perform measurement. After the measurement, the program disables all channels, disables the software connection with the E5270B, and opens a message box to confirm the end of the program. Finally, the program displays a message on the console window by clicking OK on the message box.
12	The above example is for the E5270B on the GPIB address 17. Confirm the GPIB address of your E5270B, and set the address correctly instead of “17”.

```
  Sub perform_meas(ByVal Mye5270 As Age5270)                                    '26
    Dim i As Integer = 0
    Dim j As Integer = 0
    Dim nop1 As Integer = 1
    Dim nop2 As Integer = 1
    Dim data(nop2, nop1) As String
    Dim val As String = "enter data header"
    Dim fname As String = "C:\enter_file_name.txt"
    Dim title As String = "Measurement Result"
    Dim msg As String = "No error."
    Dim err As Integer = 0

    Mye5270.SetSwitch(1, 1)        'this is dummy.                            '38
    ' insert measurement program code

    Mye5270.ErrorQuery(err, msg)                                              '41
    If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

    Mye5270.ZeroOutput(0)                                                     '44
    save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                    '47
    If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly,
"")
  End Sub                                                                     '49
```

Line	Description
26	Beginning of the perform_meas subprogram.
27 to 36	Declares variables used in this program template. The values are dummy. You must change the values to match your program. If you find unnecessary variables, delete them. *i* and *j*: Variables used to specify the element of the *data* array. *nop1* and *nop2*: Number of measurement steps. Also used to declare the *data* array. *data*: String data array used to store the measurement result data except for the header. *val*: String data variable to store the header (first line) of the measurement result data. *fname*: Full path name of the measurement result data file to be saved. *title*: Title of the message box used to display the measurement result data. *msg* and *err*: Variables used to display an error message.
38 to 39	The lines are placed as dummy. Remove the lines and insert your program code to control the instruments and perform measurement.
41 to 42	Checks if the E5270B causes an error, and goes to Check_err if an error is detected.
44 to 45	Applies 0 V from all channels and calls the save_data subprogram (lines 51 to 73).
47 to 48	Opens a message box to display error message if an error is detected.
49	End of the perform_meas subprogram.

```
  Sub save_data(ByVal fname As String, ByVal title As String, ByVal val As String,
ByVal data(,) As String, ByVal nop1 As Integer, ByVal nop2 As Integer, ByVal
Mye5270 As Age5270)                                                          '51
    Dim i As Integer = 0
    Dim j As Integer = 0
    FileOpen(1, fname, OpenMode.Output, OpenAccess.Write, OpenShare.LockReadWrite)
    Print(1, val)
    For j = 0 To nop2 - 1
      For i = 0 To nop1 - 1
        Print(1, data(j, i))
      Next
    Next
    FileClose(1)

    Dim rbx As Integer
    For j = 0 To nop2 - 1
      For i = 0 To nop1 - 1
        val = val & data(j, i)
      Next
    Next
    val = val & Chr(10) & Chr(10) & "Data save completed."
    val = val & Chr(10) & Chr(10) & "Do you want to perform measurement again?"
    rbx = MsgBox(val, vbYesNo, title)
    If rbx = vbYes Then perform_meas(Mye5270)
  End Sub                                                                    '73

End Module
```

Line	Description
51 to 73	Save_data subprogram saves measurement result data into a file specified by the *fname* variable and displays the data and a message on a message box. If Yes is clicked on the message box, calls the perform_meas subprogram again. If No is clicked, returns to the perform_meas subprogram.

High Speed Spot Measurement

Table 4-2 explains example subprogram that performs high speed spot measurement. This example measures MOSFET drain current.

Table 4-2 High-Speed Spot Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 1
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data1.txt"
 Dim title As String = "Spot Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)        'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)        'SMU3: source
 Mye5270.SetSwitch(t(1), 1)        'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)        'SMU1: drain

 Dim vd As Double = 0.5                                                       '19
 Dim vg As Double = 0.5
 Dim idcomp As Double = 0.05
 Dim igcomp As Double = 0.01
 Dim meas As Double
 Dim status As Integer
 Dim time As Double
 Mye5270.Force(t(3), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A          26
 Mye5270.Force(t(2), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A
 Mye5270.Force(t(1), 2, 2, vg, igcomp, 0)    'out= vg V, comp= igcomp A
 Mye5270.Force(t(0), 2, 2, vd, idcomp, 0)    'out= vd V, comp= idcomp A
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 17	Enables measurement channels.
19 to 25	Declares variables and sets the value.
26 to 31	Applies voltage to device and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.

```
 Mye5270.ResetTimestamp()                                                    33
 Mye5270.SpotMeas(t(0), 1, 0, meas, status)
 data(j, i) = Chr(13) & Chr(10) & meas * 1000 & ", " & time & ", " & status

 Mye5270.ZeroOutput(0)                                                      '37
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                  '40
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

Line	Description
33 to 35	Resets time stamp and performs spot measurement. And stores the measured data into the *data* variable.
37 to 38	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data1.txt file (CSV) and displays the data on a message box.
40 to 41	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Id (mA), Time (sec), Status
3.7825, 0.006, 0

Data save completed.

Do you want to perform measurement again?
```

Multi Channel Spot Measurement

Table 4-3 explains example subprogram that performs multi channel spot measurement. This example measures bipolar transistor collector and base current.

Table 4-3 Multi-Channel Spot Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 1
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Ic (mA), Time_c (sec), Status_c, Ib (mA), Time_b (sec),
Status_b, hfe"
 Dim fname As String = "C:\Agilent\data\data2.txt"
 Dim title As String = "Spot Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3}                                               '13
 Mye5270.SetSwitch(t(2), 1)        'SMU3: collector
 Mye5270.SetSwitch(t(1), 1)        'SMU2: base
 Mye5270.SetSwitch(t(0), 1)        'SMU1: emitter

 Dim vc As Double = 3                                                         '18
 Dim vb As Double = 0.7
 Dim ve As Double = 0
 Dim iccomp As Double = 0.1
 Dim ibcomp As Double = 0.01
 Dim iecomp As Double = 0.1
 Dim mch() As Integer = {t(2), t(1), 0}
 Dim mode() As Integer = {1, 1}    'current measurement
 Dim range() As Double = {0, 0}    'auto range
 Dim md(2) As Double
 Dim st(2) As Integer
 Dim tm(2) As Double
 Mye5270.Force(t(0), Age5270.ModeEnum1.VoltageOutput, 0, ve, iecomp, 0)       '30
 Mye5270.Force(t(1), Age5270.ModeEnum1.VoltageOutput, 0, vb, ibcomp, 0)
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, vc, iccomp, 0)
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 16	Enables measurement channels.
18 to 29	Declares variables and sets the value.
30 to 32	Applies voltage to device.

```
 Mye5270.ErrorQuery(err, msg)                                                    '34
 If err <> 0 Then Mye5270.ZeroOutput(0): GoTo Check_err

 Mye5270.ResetTimestamp()                                                        '37
 Mye5270.MeasureM(mch, mode, range, md, st, tm)
 data(j, i) = Chr(13) & Chr(10) & md(0) * 1000 & ", " & tm(0) & ", " & st(0)
 data(j, i) = data(j, i) & ", " & md(1) * 1000 & ", " & tm(1) & ", " & st(1)
 data(j, i) = data(j, i) & ", " & md(0) / md(1)

 Mye5270.ZeroOutput(0)                                                           '43
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                       '46
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

Line	Description
34 to 35	Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
37 to 41	Resets time stamp and performs multi channel spot measurement. And stores the measured data into the *data* variable.
43 to 44	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data2.txt file (CSV) and displays the data on a message box.
46 to 47	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Ic (mA), Time_c (sec), Status_c, Ib (mA), Time_b (sec), Status_b,
hfe
3.776, 0.0177, 0, 0.01877, 0.0202, 0, 201.172083111348

Data save completed.

Do you want to perform measurement again?
```

Pulsed Spot Measurement

Table 4-4 explains example subprogram that performs pulsed spot measurement. This example measures MOSFET drain current.

Table 4-4 Pulsed Spot Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 1
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data3.txt"
 Dim title As String = "Spot Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)         'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)         'SMU3: source
 Mye5270.SetSwitch(t(1), 1)         'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)         'SMU1: drain

 Dim vd As Double = 0.5
 Dim vg As Double = 0.5
 Dim idcomp As Double = 0.05
 Dim igcomp As Double = 0.01
 Dim base As Double = 0
 Dim width As Double = 0.001
 Dim period As Double = 0.01
 Dim hold As Double = 0.1
 Dim meas As Double
 Dim status As Integer
 Dim time As Double
 Mye5270.SetFilter(t(1), Age5270.StateEnum2.FilterOff)                        '30
 Mye5270.SetPbias(t(1), Age5270.ModeEnum3.VoltagePulse, 2, base, vg, width, period,
hold, igcomp)
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 29	Enables measurement channels. And declares variables and sets the value.
30 to 31	Sets the filter off for the pulse output channel and sets the pulse voltage source.
32 to 33	Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.

```
 Mye5270.Force(t(3), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)           '35
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)
 Mye5270.Force(t(0), Age5270.ModeEnum1.VoltageOutput, 2, vd, idcomp, 0)
 Mye5270.ResetTimestamp()
 Mye5270.MeasureP(t(0), Age5270.ModeEnum7.CurrentMeasurement, 0, meas, status,
time)

 data(j, i) = Chr(13) & Chr(10) & meas * 1000 & ", " & time & ", " & status

 Mye5270.ZeroOutput(0)                                                         '43
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                     '46
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

Line	Description
35 to 39	Applies voltage to device, resets time stamp, and performs pulsed spot measurement.
41	Stores the measured data into the *data* variable.
43 to 44	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data3.txt file (CSV) and displays the data on a message box.
46 to 47	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Id (mA), Time (sec), Status
3.89, 0.115, 0

Data save completed.

Do you want to perform measurement again?
```

Staircase Sweep Measurement

Table 4-5 explains example subprogram that performs staircase sweep measurement. This example measures MOSFET Id-Vd characteristics.

Table 4-5 Staircase Sweep Measurement Example 1

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 3
 Dim data(nop2, nop1) As String
 Dim val As String = "Vg (V), Vd (V), Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data4.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)           'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)           'SMU3: source
 Mye5270.SetSwitch(t(1), 1)           'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)           'SMU1: drain

 Dim vd1 As Double = 0                                                        '19
 Dim vd2 As Double = 3
 Dim idcomp As Double = 0.05
 Dim vg1 As Double = 1
 Dim vg2 As Double = 3
 Dim igcomp As Double = 0.01
 Dim vg As Double = vg1              'secondary sweep output value
 Dim d_vg As Double = 0              'secondary sweep step value (delta)
 If nop2 <> 1 Then d_vg = (vg2 - vg1) / (nop2 - 1)
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim p_comp As Double = 0
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double              'primary sweep output data
 Dim md(nop1) As Double              'sweep measurement data
 Dim st(nop1) As Integer             'status data at each step
 Dim tm(nop1) As Double              'time stamp data                         '36
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 17	Enables measurement channels.
19 to 36	Declares variables and sets the value.

```
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)           '38
 Mye5270.Force(t(3), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)

 For j = 0 To nop2 - 1                                                         '41
  Mye5270.SetIv(t(0), Age5270.ModeEnum2.SingleLinearV, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
  Mye5270.ErrorQuery(err, msg)
  If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  Mye5270.Force(t(1), Age5270.ModeEnum1.VoltageOutput, 0, vg, igcomp, 0)       '46
  Mye5270.ResetTimestamp()
  Mye5270.SweepIv(t(0), Age5270.ModeEnum7.CurrentMeasurement, 0, rep, sc, md, st,
tm)
  If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                        '51
   data(j, i) = Chr(13) & Chr(10) & vg & ", " & sc(i) & ", " & md(i) * 1000 & ", "
& tm(i) & ", " & st(i)
  Next

  vg = vg + d_vg
 Next

 Mye5270.ZeroOutput(0)                                                        '58
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                    '61
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly,
"")
End Sub
```

Line	Description
38 to 39	Applies voltage to device.
42	Sets the primary sweep source.
43 to 44	Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
46 to 49	Applies voltage to the device, resets time stamp, and performs staircase sweep measurement. After that, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err.
51 to 53	Stores the measured data into the *data* variable.
58 to 59	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data4.txt file (CSV) and displays the data on a message box.
61 to 63	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1, 0, 0.0020335, 0.0166, 0
1, 0.3, 3.0515, 0.0229, 0
1, 0.6, 5.6325, 0.0243, 0
1, 0.9, 7.7845, 0.0257, 0
1, 1.2, 9.6155, 0.0272, 0
1, 1.5, 11.2055, 0.0283, 0
1, 1.8, 12.63, 0.0316, 0
1, 2.1, 13.9, 0.033, 0
1, 2.4, 15.05, 0.034, 0
1, 2.7, 16.095, 0.0353, 0
1, 3, 17.045, 0.0363, 0
2, 0, 0.0025305, 0.016, 0
2, 0.3, 4.0265, 0.022, 0
2, 0.6, 7.635, 0.0236, 0
2, 0.9, 10.804, 0.0251, 0
2, 1.2, 13.565, 0.0281, 0
2, 1.5, 15.945, 0.0294, 0
2, 1.8, 18.01, 0.0305, 0
2, 2.1, 19.825, 0.0317, 0
2, 2.4, 21.445, 0.033, 0
2, 2.7, 22.915, 0.0341, 0
2, 3, 24.235, 0.0354, 0
3, 0, 0.0028565, 0.016, 0
3, 0.3, 4.8745, 0.0228, 0
3, 0.6, 9.3705, 0.0243, 0
3, 0.9, 13.445, 0.0278, 0
3, 1.2, 17.12, 0.0292, 0
3, 1.5, 20.37, 0.0302, 0
3, 1.8, 23.24, 0.0315, 0
3, 2.1, 25.75, 0.0326, 0
3, 2.4, 27.98, 0.0339, 0
3, 2.7, 29.96, 0.0352, 0
3, 3, 31.73, 0.0362, 0

Data save completed.

Do you want to perform measurement again?
```

Table 4-6 explains example subprogram that performs synchronous sweep measurement. This example uses the Mye5270.SetSweepSync function (age5270_setSweepSync function) to perform MOSFET Id-Vg measurement.

Table 4-6 Staircase Sweep Measurement Example 2

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Vg (V), Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data5.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)            'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)            'SMU3: source
 Mye5270.SetSwitch(t(1), 1)            'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)            'SMU1: drain
 Dim vpri1 As Double = 0
 Dim vpri2 As Double = 3
 Dim vsyn1 As Double = 0
 Dim vsyn2 As Double = 3
 Dim vcon1 As Double = 0
 Dim vcon2 As Double = 0
 Dim i1comp As Double = 0.01
 Dim i2comp As Double = 0.05
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim p1comp As Double = 0
 Dim p2comp As Double = 0
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double              'primary sweep output data
 Dim md(nop1) As Double              'sweep measurement data
 Dim st(nop1) As Integer             'status data at each step
 Dim tm(nop1) As Double              'time stamp data                         '35
 Mye5270.SetIv(t(1), Age5270.ModeEnum2.SingleLinearV, 0, vpri1, vpri2, nop1, hold,
delay, s_delay, i1comp, p1comp)
 Mye5270.SetSweepSync(t(0), Age5270.ModeEnum5.VoltageOutput, 0, vsyn1, vsyn2, i2comp,
p2comp)
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 35	Enables measurement channels. And declares variables and sets the value.
36 to 37	Sets the primary sweep source and the synchronous sweep source.

```
 Mye5270.ErrorQuery(err, msg)                                                    '38
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 Mye5270.Force(t(3), Age5270.ModeEnum1.VoltageOutput, 0, vcon1, 0.05, 0)         '41
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, vcon2, 0.05, 0)
 Mye5270.ResetTimestamp()
 Mye5270.SweepIv(t(0), Age5270.ModeEnum7.CurrentMeasurement, 0, rep, sc, md, st, tm)
 If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1                                                           '47
  data(j, i) = Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & tm(i) & "," & st(i)
 Next

 Mye5270.ZeroOutput(0)                                                           '51
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                       '54
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly, "")
End Sub
```

Line	Description
38 to 39	Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
41 to 45	Applies voltage to device, resets time stamp, and performs sweep measurement. After that, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err.
47 to 49	Stores the measured data into the *data* variable.
51 to 52	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data5.txt file (CSV) and displays the data on a message box.
54 to 56	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vg (V), Id (mA), Time (sec), Status
0,0.0011885,0.0186,0
0.3,2.2425,0.025,0
0.6,4.7495,0.027,0
0.9,7.5,0.0283,0
1.2,10.471,0.03,0
1.5,13.645,0.0337,0
1.8,16.995,0.0353,0
2.1,20.51,0.0365,0
2.4,24.165,0.038,0
2.7,27.95,0.0395,0
3,31.835,0.041,0

Data save completed.

Do you want to perform measurement again?
```

Table 4-7 explains example subprogram that performs synchronous sweep measurement. This example uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 4-6, MOSFET Id-Vg measurement).

Table 4-7 Staircase Sweep Measurement Example 3

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Vg (V), Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data6.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)            'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)            'SMU3: source
 Mye5270.SetSwitch(t(1), 1)            'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)            'SMU1: drain
 Dim vpri1 As Double = 0
 Dim vpri2 As Double = 3
 Dim vsyn1 As Double = 0
 Dim vsyn2 As Double = 3
 Dim vcon1 As Double = 0
 Dim vcon2 As Double = 0
 Dim i1comp As Double = 0.01
 Dim i2comp As Double = 0.05
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim p1comp As Double = 0
 Dim p2comp As Double = 0
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double              'primary sweep output data
 Dim md(nop1) As Double              'sweep measurement data
 Dim st(nop1) As Integer             'status data at each step
 Dim tm(nop1) As Double              'time stamp data                         '35
 Mye5270.SetIv(t(1), Age5270.ModeEnum2.SingleLinearV, 0, vpri1, vpri2, nop1, hold,
delay, s_delay, i1comp, p1comp)
 Mye5270.SetSweepSync(t(0), Age5270.ModeEnum5.VoltageOutput, 0, vsyn1, vsyn2, i2comp,
p2comp)
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 35	Enables measurement channels. And declares variables and sets the value.
36 to 37	Sets the primary sweep source and the synchronous sweep source.

```
 Mye5270.ErrorQuery(err, msg)                                                    '38
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 Mye5270.Force(t(3), Age5270.ModeEnum1.VoltageOutput, 0, vcon1, 0.05, 0)         '41
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, vcon2, 0.05, 0)
 Mye5270.ResetTimestamp()
 Mye5270.MsweepIv(t(0), Age5270.ModeEnum7.CurrentMeasurement, 0, rep, sc, md, st, tm)
 If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1                                                           '47
  data(j, i) = Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & tm(i) & "," & st(i)
 Next

 Mye5270.ZeroOutput(0)                                                           '51
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                       '54
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly, "")
End Sub
```

Line	Description
38 to 39	Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
41 to 45	Applies voltage to device, resets time stamp, and performs sweep measurement. After that, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err.
47 to 49	Stores the measured data into the *data* variable.
51 to 52	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data6.txt file (CSV) and displays the data on a message box.
54 to 56	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vg (V), Id (mA), Time (sec), Status
0,0.001406,0.0187,0
0.3,2.2405,0.0251,0
0.6,4.748,0.0271,0
0.9,7.5015,0.0284,0
1.2,10.473,0.0301,0
1.5,13.645,0.0339,0
1.8,17,0.0355,0
2.1,20.515,0.037,0
2.4,24.17,0.0383,0
2.7,27.955,0.0399,0
3,31.85,0.0414,0

Data save completed.

Do you want to perform measurement again?
```

Multi Channel Sweep Measurement

Table 4-8 explains example subprogram that performs multi channel sweep measurement. This example measures bipolar transistor Ic-Vb and Ib-Vb characteristics.

Table 4-8 Multi Channel Sweep Measurement Example 1

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Vb (V), Ib (mA), Ic (mA), Status_b, Status_c, Time_b (sec), Time_c
(sec)"
 Dim fname As String = "C:\Agilent\data\data7.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {2, 3, 1}                                               '13
 Mye5270.SetSwitch(t(2), 1)   'SMU1: emitter
 Mye5270.SetSwitch(t(1), 1)   'SMU3: collector
 Mye5270.SetSwitch(t(0), 1)   'SMU2: base
 Dim vc As Double = 3
 Dim ve As Double = 0
 Dim vb1 As Double = 0.3
 Dim vb2 As Double = 0.8
 Dim iccomp As Double = 0.1
 Dim ibcomp As Double = 0.001
 Dim iecomp As Double = 0.1
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim pcomp As Double = 0
 Dim mch() As Integer = {t(0), t(1), 0}   'base, collector
 Dim mode() As Integer = {1, 1}           'current measurement
 Dim range() As Double = {-0.001, -0.1}   '1 mA, 100 mA fixed range
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double                   'primary sweep output data
 Dim md(nop1 * 2) As Double               'sweep measurement data
 Dim st(nop1 * 2) As Integer              'status data at each step
 Dim tm(nop1 * 2) As Double               'time stamp data                    '35
 Mye5270.SetAdc(Age5270.AdcEnum.HspeedAdc, Age5270.ModeEnum.Auto, 1,
Age5270.AutozeroEnum.AdcZeroOff)
 Mye5270.SetAdcType(0, 0)
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 35	Enables measurement channels. And declares variables and sets the value.
36 to 37	Sets the A/D converter integration time and selects the ADC type.

```
 Mye5270.SetIv(t(0), Age5270.ModeEnum2.SingleLinearV, 0, vb1, vb2, nop1, hold, delay,
s_delay, ibcomp, pcomp)                                                          '38
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, ve, iecomp, 0)
 Mye5270.Force(t(1), Age5270.ModeEnum1.VoltageOutput, 0, vc, iccomp, 0)
 Mye5270.ResetTimestamp()
 Mye5270.SweepMiv(mch, mode, range, rep, sc, md, st, tm)
 If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1
  data(j, i) = Chr(13) & Chr(10) & sc(i) & ", " & md(2 * i) * 1000 & ", "
  data(j, i) = data(j, i) & md(2 * i + 1) * 1000 & ", " & st(2 * i) & ", " & st(2 * i + 1)
  data(j, i) = data(j, i) & ", " & tm(2 * i) & ", " & tm(2 * i + 1)
 Next

 Mye5270.ZeroOutput(0)                                                           '54
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                       '57
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly, "")
End Sub
```

Line	Description
38 to 52	Sets the sweep source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. After that, applies voltage to device, resets time stamp, and performs sweep measurement. And then, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err. Finally, stores the measured data into the *data* variable.
54 to 55	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data7.txt file (CSV) and displays the data on a message box.
57 to 59	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vb (V), Ib (mA), Ic (mA), Status_b, Status_c, Time_b (sec), Time_c
(sec)
0.3, -0.0002, 0.005, 0, 0, 0.017, 0.0178
0.35, -0.00025, 0.01, 0, 0, 0.019, 0.0195
0.4, -0.0002, 0.01, 0, 0, 0.0203, 0.0211
0.45, -0.0003, 0.01, 0, 0, 0.0219, 0.0224
0.5, -0.00025, 0.01, 0, 0, 0.0234, 0.0239
0.55, -0.00015, 0.02, 0, 0, 0.025, 0.0255
0.6, 0.00025, 0.095, 0, 0, 0.0263, 0.0271
0.65, 0.00275, 0.6, 0, 0, 0.0279, 0.0284
0.7, 0.0181, 3.72, 0, 0, 0.0294, 0.0299
0.75, 0.0873, 17.195, 0, 0, 0.031, 0.0315
0.8, 0.27085, 47.69, 0, 0, 0.0323, 0.0331

Data save completed.

Do you want to perform measurement again?
```

Table 4-9 explains example subprogram that performs multi channel sweep measurement. This example uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 4-8, bipolar transistor Ic-Vb and Ib-Vb measurement).

Table 4-9 Multi Channel Sweep Measurement Example 2

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Vb (V), Ib (mA), Ic (mA), Status_b, Status_c, Time_b (sec), Time_c
(sec)"
 Dim fname As String = "C:\Agilent\data\data8.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {2, 3, 1}                                               '13
 Mye5270.SetSwitch(t(2), 1)   'SMU1: emitter
 Mye5270.SetSwitch(t(1), 1)   'SMU3: collector
 Mye5270.SetSwitch(t(0), 1)   'SMU2: base
 Dim vc As Double = 3
 Dim ve As Double = 0
 Dim vb1 As Double = 0.3
 Dim vb2 As Double = 0.8
 Dim iccomp As Double = 0.1
 Dim ibcomp As Double = 0.001
 Dim iecomp As Double = 0.1
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim pcomp As Double = 0
 Dim mch() As Integer = {t(0), t(1), 0}   'base, collector
 Dim mode() As Integer = {1, 1}           'current measurement
 Dim range() As Double = {-0.001, -0.1}   '1 mA, 100 mA fixed range
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double                   'primary sweep output data
 Dim md(nop1 * 2) As Double               'sweep measurement data
 Dim st(nop1 * 2) As Integer              'status data at each step
 Dim tm(nop1 * 2) As Double               'time stamp data                    '35
 Mye5270.SetAdc(Age5270.AdcEnum.HspeedAdc, Age5270.ModeEnum.Auto, 1,
Age5270.AutozeroEnum.AdcZeroOff)
 Mye5270.SetAdcType(0, 0)
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 35	Enables measurement channels. And declares variables and sets the value.
36 to 37	Sets the A/D converter integration time and selects the ADC type.

```
 Mye5270.SetIv(t(0), Age5270.ModeEnum2.SingleLinearV, 0, vb1, vb2, nop1, hold, delay,
s_delay, ibcomp, pcomp)                                                          '38
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, ve, iecomp, 0)
 Mye5270.Force(t(1), Age5270.ModeEnum1.VoltageOutput, 0, vc, iccomp, 0)
 Mye5270.ResetTimestamp()
 Mye5270.MsweepMiv(mch, mode, range, rep, sc, md, st, tm)
 If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1
  data(j, i) = Chr(13) & Chr(10) & sc(i) & ", " & md(2 * i) * 1000 & ", "
  data(j, i) = data(j, i) & md(2 * i + 1) * 1000 & ", " & st(2 * i) & ", " & st(2 * i + 1)
  data(j, i) = data(j, i) & ", " & tm(2 * i) & ", " & tm(2 * i + 1)
 Next

 Mye5270.ZeroOutput(0)                                                           '54
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                       '57
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly, "")
End Sub
```

Line	Description
38 to 52	Sets the sweep source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. After that, applies voltage to device, resets time stamp, and performs sweep measurement. And then, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err. Finally, stores the measured data into the *data* variable.
54 to 55	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data8.txt file (CSV) and displays the data on a message box.
57 to 59	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vb (V), Ib (mA), Ic (mA), Status_b, Status_c, Time_b (sec), Time_c
(sec)
0.3, -0.00025, 0.01, 0, 0, 0.0159, 0.0159
0.35, -0.0002, 0.01, 0, 0, 0.0175, 0.0175
0.4, -0.00025, 0.01, 0, 0, 0.0185, 0.0185
0.45, -0.00025, 0, 0, 0, 0.0198, 0.0198
0.5, -0.0002, 0.01, 0, 0, 0.0208, 0.0208
0.55, -0.0001, 0.02, 0, 0, 0.0221, 0.0221
0.6, 0.0002, 0.095, 0, 0, 0.0231, 0.0231
0.65, 0.00265, 0.59, 0, 0, 0.0243, 0.0243
0.7, 0.01785, 3.67, 0, 0, 0.0256, 0.0256
0.75, 0.08685, 16.935, 0, 0, 0.0266, 0.0266
0.8, 0.27075, 47.22, 0, 0, 0.0279, 0.0279

Data save completed.

Do you want to perform measurement again?
```

Pulsed Sweep Measurement

Table 4-10 explains example subprogram that performs pulsed sweep measurement. This example measures bipolar transistor Ic-Vc characteristics.

Table 4-10 Pulsed Sweep Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 3
 Dim data(nop2, nop1) As String
 Dim val As String = "Ib (uA), Vc (V), Ic (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data9.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {3, 2, 1}                                               '13
 Mye5270.SetSwitch(t(2), 1)        'SMU1: emitter
 Mye5270.SetSwitch(t(1), 1)        'SMU2: base
 Mye5270.SetSwitch(t(0), 1)        'SMU3: collector

 Dim vc1 As Double = 0
 Dim vc2 As Double = 3
 Dim iccomp As Double = 0.05
 Dim ib1 As Double = 0.00005       ' 50 uA
 Dim ib2 As Double = 0.00015       '150 uA
 Dim vbcomp As Double = 5
 Dim ibo As Double = ib1           'secondary sweep output value
 Dim d_ib As Double = 0            'secondary sweep step value (delta)
 If nop2 <> 1 Then d_ib = (ib2 - ib1) / (nop2 - 1)
 Dim hold As Double = 0.1
 Dim width As Double = 0.001
 Dim period As Double = 0.01
 Dim base As Double = 0

 Dim rep As Integer = nop1
 Dim sc(nop1) As Double              'primary sweep output data
 Dim md(nop1) As Double              'sweep measurement data
 Dim st(nop1) As Integer             'status data at each step
 Dim tm(nop1) As Double              'time stamp data                         '36
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 16	Enables measurement channels.
18 to 36	Declares variables and sets the value.

```
 Mye5270.SetFilter(t(0), Age5270.StateEnum2.FilterOff)                          '38
 Mye5270.SetAdc(Age5270.AdcEnum.HspeedAdc, Age5270.ModeEnum.Auto, 1, 0)
 Mye5270.SetAdcType(0, 0)
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)

 For j = 0 To nop2 - 1                                                          '43
  Mye5270.SetPiv(t(0), 1, 0, base, vc1, vc2, nop1, hold, width, period, iccomp)
  Mye5270.ErrorQuery(err, msg)
  If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  Mye5270.Force(t(1), Age5270.ModeEnum1.CurrentOutput, 0, ibo, vbcomp, 0)       '48
  Mye5270.ResetTimestamp()
  Mye5270.SweepPiv(t(0), 1, 0, rep, sc, md, st, tm)
  If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                         '53
   data(j, i) = Chr(13) & Chr(10) & ibo * 1000000 & "," & sc(i) & "," & md(i) * 1000
& "," & tm(i) & "," & st(i)
  Next

  ibo = ibo + d_ib
 Next

 Mye5270.ZeroOutput(0)                                                          '60
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                      '63
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly,
"")
End Sub
```

Line	Description
38 to 41	Sets the filter off for the pulse source, sets the A/D converter, and applies voltage to device.
44 to 46	Sets the primary sweep source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
48 to 51	Applies voltage to the device, resets time stamp, and performs pulsed sweep measurement. After that, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err.
53 to 55	Stores the measured data into the *data* variable.
60 to 61	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data9.txt file (CSV) and displays the data on a message box.
63 to 65	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Ib (uA), Vc (V), Ic (mA), Time (sec), Status
50,0,-0.045,0.1157,0
50,0.3,8.95,0.1257,0
50,0.6,9.735,0.1357,0
50,0.9,9.8,0.1457,0
50,1.2,9.82,0.1557,0
50,1.5,9.835,0.1657,0
50,1.8,9.885,0.1757,0
50,2.1,9.91,0.1857,0
50,2.4,9.915,0.1957,0
50,2.7,9.93,0.2057,0
50,3,9.96,0.2157,0
100,0,-0.1,0.1148,0
100,0.3,15.645,0.1248,0
100,0.6,18.155,0.1348,0
100,0.9,18.76,0.1448,0
100,1.2,18.945,0.1548,0
100,1.5,18.98,0.1648,0
100,1.8,19.08,0.1748,0
100,2.1,19.165,0.1848,0
100,2.4,19.205,0.1948,0
100,2.7,19.24,0.2048,0
100,3,19.305,0.2148,0
150,0,-0.145,0.1136,0
150,0.3,20.9,0.1236,0
150,0.6,24.55,0.1336,0
150,0.9,26.52,0.1436,0
150,1.2,27.34,0.1536,0
150,1.5,27.61,0.1636,0
150,1.8,27.79,0.1736,0
150,2.1,27.84,0.1836,0
150,2.4,27.965,0.1936,0
150,2.7,28.015,0.2036,0
150,3,28.13,0.2136,0

Data save completed.

Do you want to perform measurement again?
```

Staircase Sweep with Pulsed Bias Measurement

Table 4-11 explains example subprogram that performs staircase sweep with pulsed bias measurement. This example measures MOSFET Id-Vd characteristics.

Table 4-11 Staircase Sweep with Pulsed Bias Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 11
 Dim nop2 As Integer = 3
 Dim data(nop2, nop1) As String
 Dim val As String = "Vg (V), Vd (V), Id (mA), Time (sec), Status"
 Dim fname As String = "C:\Agilent\data\data10.txt"
 Dim title As String = "Sweep Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0
 Dim t() As Integer = {1, 2, 3, 4}                                            '12
 Mye5270.SetSwitch(t(3), 1)           'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)           'SMU3: source
 Mye5270.SetSwitch(t(1), 1)           'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)           'SMU1: drain
 Dim vd1 As Double = 0                                                        '17
 Dim vd2 As Double = 3
 Dim idcomp As Double = 0.05
 Dim vg1 As Double = 1
 Dim vg2 As Double = 3
 Dim igcomp As Double = 0.01
 Dim vg As Double = vg1              'secondary sweep output value
 Dim d_vg As Double = 0              'secondary sweep step value (delta)
 If nop2 <> 1 Then d_vg = (vg2 - vg1) / (nop2 - 1)
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim s_delay As Double = 0
 Dim p_comp As Double = 0
 Dim width As Double = 0.001
 Dim period As Double = 0.01
 Dim p_hold As Double = 0.1
 Dim rep As Integer = nop1
 Dim sc(nop1) As Double              'primary sweep output data
 Dim md(nop1) As Double              'sweep measurement data
 Dim st(nop1) As Integer             'status data at each step
 Dim tm(nop1) As Double              'time stamp data                         '37
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
12 to 16	Enables measurement channels.
17 to 37	Declares variables and sets the value.

```
 Mye5270.SetFilter(t(1), Age5270.StateEnum2.FilterOff)                          '39
 Mye5270.Force(t(3), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)
 Mye5270.Force(t(2), Age5270.ModeEnum1.VoltageOutput, 0, 0, 0.05, 0)

 For j = 0 To nop2 - 1                                                          '43
  Mye5270.SetPbias(t(1), Age5270.ModeEnum3.VoltagePulse, 0, 0, vg, width, period,
p_hold, igcomp)
  Mye5270.SetIv(t(0), Age5270.ModeEnum2.SingleLinearV, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
  Mye5270.ErrorQuery(err, msg)
  If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  Mye5270.ResetTimestamp()                                                      '49
  Mye5270.SweepPbias(t(0), Age5270.ModeEnum7.CurrentMeasurement, 0, rep, sc, md,
st, tm)
  If rep <> nop1 Then Mye5270.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                         '53
   data(j, i) = Chr(13) & Chr(10) & vg & ", " & sc(i) & ", " & md(i) * 1000 & ", "
& tm(i) & ", " & st(i)
  Next

  vg = vg + d_vg
 Next

 Mye5270.ZeroOutput(0)                                                          '60
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                      '63
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If rep <> nop1 Then MsgBox("No. of data: " & rep & " (not " & nop1 & ")", vbOKOnly,
"")
End Sub
```

Line	Description
39 to 41	Applies voltage to device.
43 to 47	Sets the pulse source and the primary sweep source. And checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.
49 to 51	Resets time stamp and performs sweep measurement. After that, checks the number of returned data. If it was not rep= nop1, forces 0 V and goes to Check_err.
53 to 55	Stores the measured data into the *data* variable.
60 to 61	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data10.txt file (CSV) and displays the data on a message box.
63 to 65	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1, 0, 0.01, 0.1179, 0
1, 0.3, 3.08, 0.1279, 0
1, 0.6, 5.665, 0.1379, 0
1, 0.9, 7.82, 0.1479, 0
1, 1.2, 9.645, 0.1579, 0
1, 1.5, 11.24, 0.1679, 0
1, 1.8, 12.65, 0.1779, 0
1, 2.1, 13.915, 0.1879, 0
1, 2.4, 15.06, 0.1979, 0
1, 2.7, 16.095, 0.2079, 0
1, 3, 17.035, 0.2179, 0
2, 0, 0.015, 0.1139, 0
2, 0.3, 4.055, 0.1239, 0
2, 0.6, 7.695, 0.1339, 0
2, 0.9, 10.87, 0.1439, 0
2, 1.2, 13.625, 0.1539, 0
2, 1.5, 16, 0.1639, 0
2, 1.8, 18.065, 0.1739, 0
2, 2.1, 19.875, 0.1839, 0
2, 2.4, 21.485, 0.1939, 0
2, 2.7, 22.94, 0.2039, 0
2, 3, 24.265, 0.2139, 0
3, 0, 0.025, 0.1145, 0
3, 0.3, 4.915, 0.1245, 0
3, 0.6, 9.44, 0.1345, 0
3, 0.9, 13.535, 0.1445, 0
3, 1.2, 17.215, 0.1545, 0
3, 1.5, 20.47, 0.1645, 0
3, 1.8, 23.34, 0.1745, 0
3, 2.1, 25.86, 0.1845, 0
3, 2.4, 28.075, 0.1945, 0
3, 2.7, 30.045, 0.2045, 0
3, 3, 31.805, 0.2145, 0

Data save completed.

Do you want to perform measurement again?
```

Breakdown Voltage Measurement

Table 4-12 explains example subprogram that performs quasi pulsed spot measurement. This example measures bipolar transistor breakdown voltage.

Table 4-12 Breakdown Voltage Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 1
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Vbd (V), Status"
 Dim fname As String = "C:\Agilent\data\data11.txt"
 Dim title As String = "Vbd Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3}                                               '13
 Mye5270.SetSwitch(t(2), 1)        'SMU3: collector
 Mye5270.SetSwitch(t(1), 1)        'SMU2: base - open
 Mye5270.SetSwitch(t(0), 1)        'SMU1: emitter

 Dim vstart As Double = 0                                                     '18
 Dim vstop As Double = 100
 Dim vb As Double = 0.7
 Dim ve As Double = 0
 Dim iccomp As Double = 0.005
 Dim ibcomp As Double = 0.01
 Dim iecomp As Double = 0.1
 Dim hold As Double = 0
 Dim delay As Double = 0
 Dim meas As Double
 Dim status As Long

 Mye5270.SetBdv(t(2), 0, vstart, vstop, iccomp, hold, delay)                  '30
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 16	Enables measurement channels.
18 to 28	Declares variables and sets the value.
30 to 32	Sets the quasi pulse source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.

```
 Mye5270.Force(t(0), Age5270.ModeEnum1.VoltageOutput, 0, ve, iecomp, 0)        '34
 Mye5270.MeasureBdv(Age5270.IntervalEnum.Short, meas, status)
 data(j, i) = Chr(13) & Chr(10) & meas & ", " & status

 Mye5270.ZeroOutput(0)                                                         '38
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                     '41
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

Line	Description
34 to 36	Applies voltage to device and performs quasi pulsed spot measurement. And stores the measured data into the *data* variable.
38 to 39	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data11.txt file (CSV) and displays the data on a message box.
41 to 42	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Vbd (V), Status
55.86, 8

Data save completed.

Do you want to perform measurement again?
```

Leakage Current Measurement

Table 4-13 explains example subprogram that performs quasi pulsed spot measurement. This example measures MOSFET drain current.

Table 4-13 Leakage Current Measurement Example

```
Sub perform_meas(ByVal Mye5270 As Age5270)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 1
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Id (mA), Status"
 Dim fname As String = "C:\Agilent\data\data12.txt"
 Dim title As String = "Leak Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3, 4}                                            '13
 Mye5270.SetSwitch(t(3), 1)         'SMU4: substrate
 Mye5270.SetSwitch(t(2), 1)         'SMU3: source
 Mye5270.SetSwitch(t(1), 1)         'SMU2: gate
 Mye5270.SetSwitch(t(0), 1)         'SMU1: drain

 Dim vd As Double = 5                                                         '19
 Dim vg As Double = 0
 Dim start As Double = -5
 Dim idcomp As Double = 0.05
 Dim igcomp As Double = 0.01
 Dim hold As Double = 0.1
 Dim delay As Double = 0.001
 Dim meas As Double
 Dim status As Integer

 Mye5270.SetIleak(t(0), 0, vd, idcomp, start, hold, delay)                    '29
 Mye5270.ErrorQuery(err, msg)
 If err <> 0 Then Mye5270.ZeroOutput(0) : GoTo Check_err
```

Line	Description
2 to 11	Declares variables used in the program template. And sets the proper values.
13 to 17	Enables measurement channels.
19 to 27	Declares variables and sets the value.
29 to 31	Sets the quasi pulse source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err.

```
 Mye5270.Force(t(3), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A            33
 Mye5270.Force(t(2), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A
 Mye5270.Force(t(1), 2, 2, vg, igcomp, 0)    'out= vg V, comp= igcomp A

 Mye5270.SpotMeas(t(0), 1, 0, meas, status)         'current measurement
 data(j, i) = Chr(13) & Chr(10) & meas * 1000 & ", " & time & ", " & status

 Mye5270.ZeroOutput(0)                                                        '40
 save_data(fname, title, val, data, nop1, nop2, Mye5270)

Check_err:                                                                    '43
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

Line	Description
33 to 38	Applies voltage to device and performs quasi pulsed spot measurement. And stores the measured data into the *data* variable.
40 to 41	Applies 0 V from the all channels. And transfers the data stored in the *data* variable to the save_data subprogram (see Table 4-1). And the subprogram will save the data into the C:\Agilent\data\data12.txt file (CSV) and displays the data on a message box.
43 to 44	Displays a message box to show an error message if the error is detected.

Measurement Result Example

```
Id (mA), Status
12.175, 0

Data save completed.

Do you want to perform measurement again?
```

5 Programming Examples for C++ Users

This chapter provides programming examples to perform the following measurements using the Agilent E5260/E5270 and the E5260/E5270 VXI*plug&play* driver.

- “Programming Basics”
- “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”

NOTE

About Program Code

Programming examples are provided as a subprogram that can be run with the project template shown in Table 5-1. To execute the program, insert the subprogram instead of the perform_meas subprogram in the template.

NOTE

To Start Program

If you create the measurement program by modifying the example code shown in Table 5-1, the program can be run by clicking the Run button on the Visual C++ Main window.

NOTE

For the Agilent E5260 Users

The example program code uses the Agilent E5270 VXI*plug&play* driver. So the modification is required for the Agilent E5260.

At first, delete age5270_asu, age5270_asuLed, and age5270_setAdcType. There is no replaceable function for them. Second, change the string age5270 to age5260. And correct the parameter values for some functions. Available values are different for each SMU. See “Parameters” on page 2-7.

Finally, correct the syntax of the age5260_setAdc function. See “age5260_setAdc” on page 2-38.

Programming Basics

This section provides the basic information for programming using the Agilent E5260/E5270 VXI*plug&play* driver.

- "To Create Your Project Template"
- "To Create Measurement Program"

To Create Your Project Template

This section explains how to create a project template in the C language. Before starting programming, create your project template, and keep it as your reference. It will remove the conventional task in the future programming.

Step 1. Connect instrument (e.g. Agilent E5270) to computer via GPIB.

Step 2. 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 3. 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. \Program Files\VISA\winnt\include) that stores the age52*x*0.h file and the VISA related include files
2. Additional library search path:
   - 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 age52*x*0.lib file and the VISA related library files
3. Additional project link library:
   - age52*x*0.lib

where, 52*x*0 is 5260 for the Agilent E5260 or 5270 for the Agilent E5270.

Step 4. Open a source file (.cpp) in the project, and enter a program code as template. See Table 5-1 for example. The program code is written in Microsoft Visual C++.

Step 5. Save the project as your template (e.g. \test\my_temp).

Table 5-1 Example Template Program Code for Visual C++

```
#include   <stdio.h>                                                    /* 1 */
#include   <stdlib.h>
#include   <visa.h>
#include   "age5270.h"

void check_err (ViSession vi, ViStatus ret) {                          /* 6 */
 ViInt32    inst_err;
 ViChar     err_msg[256];

 if(VI_SUCCESS > ret) {
    if ( age5270_INSTR_ERROR_DETECTED == ret ) {
       age5270_error_query(vi, &inst_err, err_msg);
       printf("Instrument Error: %ld\n %s\n", inst_err, err_msg);
      }
    else {
       age5270_error_message(vi, ret, err_msg);
       printf("Driver Error: %ld\n %s\n", ret, err_msg);
      }
   }
 }                                                                     /* 20 */

void perform_meas (ViSession vi, ViStatus ret) {                       /* 22 */
     /* insert program code */
}

ViStatus main ( )                                                      /* 26 */
{
 ViStatus    ret;                                                      /* 28 */
 ViSession   vi;
 ViChar      err_msg[256];                                             /* 30 */
```

Line	Description
1 to 4	Required to use the Agilent E5260/E5270 VXI*plug&play* driver. 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).
6 to 20	Checks if the passed “ret” value indicates normal status, and returns to the line that called this subprogram. If the value indicates an instrument error status or a device error status, the error message will be displayed.
22 to 24	Complete the perform_meas subprogram to perform measurement.
26	Beginning of the main program.
28 to 30	Declares variables used in the main program.

```
/* Starting the session */
 ret = age5270_init("GPIB::17::INSTR", VI_TRUE, VI_TRUE, &vi);            /* 33 */
 if ( ( ret < VI_SUCCESS ) || ( vi == VI_NULL ) ) {
    printf("Initialization failure.\n Status code: %d.\n", ret);
    if ( vi != VI_NULL ) {
       age5270_error_message(vi, ret, err_msg);
       printf("Error: %ld\n %s\n", ret, err_msg);
      }
    exit (ret);
   }                                                                        /* 41 */

 ret = age5270_reset(vi);                          /* resets E5270           43 */
 ret = age5270_timeOut(vi, 60000);                 /* sets 60 second timeout    */
 ret = age5270_errorQueryDetect(vi, VI_TRUE);      /* turns on error detection  */

 perform_meas(vi, ret);             /* calls perform_meas subprogram          47 */
 /* ret = age5270_cmd(vi, "aa");       sends an invalid command                 */
 /* check_err(vi, ret);                checks check_err subprogram operation    */

 /* Closing the session
 ret = age5270_close(vi);                                                   52 */
 check_err(vi, ret);

 return VI_SUCCESS;                                                         /* 55 */
}
```

Line	Description
33	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”.
34 to 41	Checks the status returned by the age5270_init function. Displays the error message and stops the program execution if an error status is returned.
43 to 45	Resets the Agilent E5270B, sets the driver I/O time out to 60 seconds, and enables the automatic instrument error checking.
47	Calls the perform_meas subprogram (line 22).
48 to 49	Should be deleted or commented out before executing the program. The lines are just used to check the operation of the check_err subprogram.
52	Disables the software connection with the Agilent E5270B.
53	Calls the check_err subprogram to check if an error status is returned for the line 52.
55 to 56	End of the main program.

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 5-1, and complete the perform_meas subprogram. Then use the Agilent E5260/E5270 VXI*plug&play* driver functions:

- age52*x*0_setSwitch to enable/disable the source/measurement channels
- age52*x*0_force, age52*x*0_setIv, etc. to set source outputs
- age52*x*0_spotMeas, age52*x*0_sweepIv, etc. to perform measurements
- age52*x*0_zeroOutput to disable source outputs

where, age52*x*0_ is age5260_ for the Agilent E5260 driver or age5270_ for the Agilent E5270 driver.

Step 7. Insert the code to display, store, or calculate data into the subprogram.

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

High Speed Spot Measurement

Table 5-2 explains an example subprogram that performs the high speed spot measurement. The following subprogram will apply voltage to a MOSFET, measure drain current, and display the measurement result data.

Table 5-2 High Speed Spot Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32   drain;                                        /* 3 */
ViInt32   gate;
ViInt32   source;
ViInt32   bulk;

drain =   1;   /* SMU1 */
gate =    2;   /* SMU2 */
source =  4;   /* SMU4 */
bulk =    6;   /* SMU6 */                             /* 11 */

ret = age5270_setSwitch(vi, drain, 1);                  /* 13 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 17 */

ViReal64   vd;                                          /* 19 */
ViReal64   vg;
ViReal64   idcomp;
ViReal64   igcomp;
ViReal64   meas;
ViInt32    status;

vd =       1.5;
idcomp =   0.05;
vg =       1.5;
igcomp =   0.01;                                        /* 29 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 11	Declares variables, and defines the value.
13 to 16	Enables measurement channels.
17	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
19 to 29	Declares variables, and defines the value.

```
ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);                /* 31 */
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, gate, age5270_VF_MODE, 2, vg, igcomp, 0);
ret = age5270_force(vi, drain, age5270_VF_MODE, 2, vd, idcomp, 0);
check_err (vi, ret);                                                         /* 35 */

ret = age5270_spotMeas(vi,drain,age5270_IM_MODE,0,&meas,&status,0);
check_err (vi, ret);                                                         /* 38 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                                /* 40 */
check_err (vi, ret);                                                         /* 41 */

printf("Id = %9.6f mA (at %3.1f V)\n", meas * 1000, vd);                     /* 43 */
printf("Vg = %3.1f V\n", vg);

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);                              /* 46 */
check_err (vi, ret);                                                         /* 47 */
}
```

Line	Description
31 to 34	Applies voltage to device.
37	Performs high speed spot measurement for the drain terminal.
40	Sets the specified port to the zero output state.
43 to 44	Displays the measurement result data.
46	Disables all ports.
35, 38, 41, and 47	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
48	End of the perform_meas subprogram.

Measurement Result Example

```
Id = 14.255001 mA (at 1.5 V)
Vg = 1.5 V
```

Multi Channel Spot Measurement

Table 5-3 explains an example subprogram that performs the multi channel spot measurement. The following subprogram will apply voltage to a bipolar transistor, measure Ic and Ib, calculate hfe value, and display the measurement result data.

Table 5-3 Multi Channel Spot Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32     emitter;                                    /* 3 */
ViInt32     base;
ViInt32     collector;
emitter =   1;  /* SMU1 */
base =      2;  /* SMU2 */
collector = 4;  /* SMU4 */                              /* 8 */

ret = age5270_setSwitch(vi, emitter, 1);               /* 10 */
ret = age5270_setSwitch(vi, base, 1);
ret = age5270_setSwitch(vi, collector, 1);
check_err (vi, ret);                                   /* 13 */

ViReal64    vc;                                        /* 15 */
ViReal64    vb;
ViReal64    iccomp;
ViReal64    ibcomp;
vc =     3;
iccomp = 1;
vb =     0.7;
ibcomp = 0.01;

ViInt32     mch[3];
ViInt32     mode[2];
ViReal64    range[2];
ViReal64    md[2];
ViInt32     st[2];
ViReal64    tm[2];                                     /* 29 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 8	Declares variables, and defines the value.
10 to 12	Enables measurement channels.
13	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
15 to 29	Declares variables, and defines the value.

```
mch[0] =    collector;                                                  /* 31 */
mch[1] =    base;
mch[2] =    0;
mode[0] =   1;
mode[1] =   1;
range[0] =  0;
range[1] =  0;
ret = age5270_resetTimestamp(vi);                                       /* 38 */
ret = age5270_force(vi, emitter, age5270_VF_MODE, 0, 0, 0.2, 0);
ret = age5270_force(vi, base, age5270_VF_MODE, 0, vb, ibcomp, 0);
ret = age5270_force(vi, collector, age5270_VF_MODE, 0, vc, iccomp, 0);
check_err (vi, ret);                                                    /* 42 */

ret = age5270_measureM(vi, mch, mode, range, &md[0], &st[0], &tm[0]);
check_err (vi, ret);                                                    /* 45 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                           /* 47 */
check_err (vi, ret);                                                    /* 48 */

printf("Ic = %8.6f mA (Time: %8.6f sec)\n", md[0] * 1000, tm[0]);
printf("Ib = %8.6f mA (Time: %8.6f sec)\n", md[1] * 1000, tm[1]);
printf("hfe = %10.6f \n", md[0]/md[1]);                                  /* 52 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);                         /* 54 */
check_err (vi, ret);                                                    /* 55 */
}
```

Line	Description
31 to 37	Defines the value for the variables used for the measurement channel.
38	Resets time stamp.
39 to 41	Applies voltage to device.
44	Performs multi channel spot measurement.
47	Sets the specified port to the zero output state.
50 to 52	Displays the measurement result data.
54	Disables all ports.
42, 45, 48 and 55	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
56	End of the perform_meas subprogram.

Measurement Result Example

```
Ic = 3.846500 mA (Time: 0.093200 sec)
Ib = 0.018970 mA (Time: 0.094300 sec)
hfe = 202.767528
```

Pulsed Spot Measurement

Table 5-4 explains an example subprogram that performs the pulsed spot measurement. The following subprogram will apply voltage to a MOSFET, measure drain current, and display the measurement result data.

Table 5-4 Pulsed Spot Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32   drain;                                        /* 3 */
ViInt32   gate;
ViInt32   source;
ViInt32   bulk;

drain =  1;  /* SMU1 */
gate =   2;  /* SMU2 */
source = 4;  /* SMU4 */
bulk =   6;  /* SMU6 */                                 /* 11 */

ret = age5270_setSwitch(vi, drain, 1);                  /* 13 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 17 */

ViReal64  vd;                                           /* 19 */
ViReal64  vg;
ViReal64  idcomp;
ViReal64  igcomp;
ViReal64  base;
ViReal64  width;
ViReal64  period;
ViReal64  hold;
ViReal64  meas;
ViInt32   status;                                       /* 28 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 11	Declares variables, and defines the value.
13 to 16	Enables measurement channels.
17	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
19 to 28	Declares variables, and defines the value.

```
vd =     1.5;                                                          /* 30 */
idcomp = 0.05;
vg =     1.5;
igcomp = 0.01;
base =   0;
width =  0.001;
period = 0.01;
hold =   0.1;                                                          /* 37 */

ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_setPbias(vi, gate, age5270_VF_MODE, 2, base, vg,
width, period, hold, igcomp);
ret = age5270_force(vi, drain, age5270_VF_MODE, 2, vd, idcomp, 0);
check_err (vi, ret);                                                   /* 43 */

ret = age5270_measureP(vi, drain, age5270_IM_MODE, 0, &meas, &status, 0);
check_err (vi, ret);                                                   /* 46 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);
check_err (vi, ret);                                                   /* 49 */

printf("Id = %9.6f mA (at %3.1f V)\n", meas * 1000, vd);
printf("Vg = %3.1f V\n", vg);                                          /* 52 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
check_err (vi, ret);                                                   /* 55 */
}
```

Line	Description
30 to 37	Defines the variable values for the source channels.
39 to 42	Applies voltage to device, and sets the pulsed bias source.
45	Performs pulsed spot measurement.
48	Sets the specified port to the zero output state.
51 to 52	Displays the measurement result data.
54	Disables all ports.
43, 46, 49 and 55	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
56	End of the perform_meas subprogram.

Measurement Result Example

```
Id = 14.255000 mA (at 1.5 V)
Vg = 1.5 V
```

Staircase Sweep Measurement

Table 5-5 explains an example subprogram that performs the staircase sweep measurement. The following subprogram performs I-V measurement and save the measurement results (MOSFET Id-Vd characteristics) into a file.

Table 5-5 Staircase Sweep Measurement Example 1

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32   drain =     1;  /* SMU1 */                    /* 4 */
ViInt32   gate =      2;  /* SMU2 */
ViInt32   source =    4;  /* SMU4 */
ViInt32   bulk =      6;  /* SMU6 */

ViReal64  vd =        3;
ViReal64  vg =        3;
ViReal64  idcomp =    0.05;
ViReal64  igcomp =    0.01;
ViReal64  hold =      0;
ViReal64  delay =     0;
ViReal64  s_delay =   0;
ViReal64  p_comp =    0;

ViInt32   nop1 =      11;
ViInt32   nop2 =      3;

ViInt32   rep;
ViReal64  sc[33];
ViReal64  md[33];
ViInt32   st[33];
ViReal64  tm[33];
ViReal64  dvg[3];

ViInt32   i =         0;
ViInt32   j;
ViInt32   n;

ViChar    f_name[] = "C:\Agilent\data\data1.txt";
ViChar    head1[] =  "Vg (V), Vd (V), Id (mA), Time (sec), Stat
us";
ViChar    msg1[] =   "Saving data...";
ViChar    msg2[] =   "Data save completed.";
ViChar    c =        '\n';                              /* 36 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 36	Declares variables, and defines the value.

```
ret = age5270_setSwitch(vi, drain, 1);                                    /* 38 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                                      /* 42 */

ret = age5270_resetTimestamp(vi);                                         /* 44 */
ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                               /* 48 */
   dvg[j] = (j + 1) * vg / nop2;
   ret = age5270_force(vi, gate, age5270_VF_MODE, 0, dvg[j], igcomp, 0);
   ret = age5270_setIv(vi, drain, age5270_SWP_VF_SGLLIN, 0, 0, vd, nop1, hold,
delay, s_delay, idcomp, p_comp);
   check_err (vi, ret);

   ret = age5270_sweepIv(vi, drain, age5270_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[i], &tm[i]);
   check_err (vi, ret);

   if ( rep = nop1 ) {
      i = i + nop1;
    }
   else {
      printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep,
nop1);
      ret = age5270_zeroOutput(vi, age5270_CH_ALL);
      ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                        /* 67 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                             /* 69 */
check_err (vi, ret);
```

Line	Description
38 to 41	Enables measurement channels.
44	Resets time stamp.
45 to 46	Applies voltage to device.
48 to 67	Applies dc voltage and sweep voltage, and performs staircase sweep measurement. After that, disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
69	Sets the specified port to the zero output state.
42 and 70	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
printf(" Vg (V),  Vd (V),  Id (mA)\n");                                    /* 72 */

for (j = 0; j < nop2; j++){
   n = j * nop1;
   for (i = n; i < n + nop1; i++){
      printf(" %4.2f,    %4.2f,    %9.6f \n", dvg[j], sc[i], md[i] * 1000);
    }
 }                                                                         /* 79 */

FILE *stream;                                                              /* 81 */

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 (j = 0; j < nop2; j++){
      n = j * nop1;
      for (i = n; i < n + nop1; i++){
         fprintf( stream, "%4.2f, %4.2f, %9.6f, %8.6f, %d\n", dvg[j], sc[i], md[i]
* 1000, tm[i], st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                         /* 100 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);                            /* 102 */
check_err (vi, ret);

}
```

Line	Description
72 to 79	Displays the measurement result data.
81 to 100	Saves the measurement results into a file (C:\Agilent\data\data1.txt, CSV file).
102	Disables all ports.
103	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
105	End of the perform_meas subprogram.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1.00, 0.00, -0.000114, 0.072100, 0
1.00, 0.30,  3.180000, 0.090500, 0
1.00, 0.60,  5.850000, 0.092300, 0
1.00, 0.90,  8.085500, 0.093500, 0
1.00, 1.20,  9.972000, 0.094900, 0
1.00, 1.50, 11.625000, 0.098300, 0
1.00, 1.80, 13.085000, 0.099300, 0
1.00, 2.10, 14.410000, 0.100600, 0
1.00, 2.40, 15.595000, 0.101600, 0
1.00, 2.70, 16.690000, 0.102900, 0
1.00, 3.00, 17.680000, 0.104300, 0
2.00, 0.00, -0.000117, 0.202400, 0
2.00, 0.30,  4.168000, 0.220800, 0
2.00, 0.60,  7.882000, 0.222300, 0
2.00, 0.90, 11.150500, 0.223800, 0
2.00, 1.20, 13.975000, 0.227200, 0
2.00, 1.50, 16.425000, 0.228300, 0
2.00, 1.80, 18.540000, 0.229600, 0
2.00, 2.10, 20.420000, 0.230600, 0
2.00, 2.40, 22.080000, 0.231800, 0
2.00, 2.70, 23.580000, 0.233100, 0
2.00, 3.00, 24.950000, 0.234200, 0
3.00, 0.00, -0.000114, 0.333300, 0
3.00, 0.30,  5.028500, 0.351800, 0
3.00, 0.60,  9.638000, 0.353500, 0
3.00, 0.90, 13.825000, 0.357000, 0
3.00, 1.20, 17.570000, 0.358000, 0
3.00, 1.50, 20.905000, 0.359300, 0
3.00, 1.80, 23.830000, 0.360300, 0
3.00, 2.10, 26.405000, 0.361700, 0
3.00, 2.40, 28.670000, 0.363000, 0
3.00, 2.70, 30.695000, 0.364000, 0
3.00, 3.00, 32.505000, 0.365300, 0
```

Table 5-6 uses the age5270_setSweepSync function to perform MOSFET Id-Vg measurement.

Table 5-6 Staircase Sweep Measurement Example 2

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32   drain =    1;  /* SMU1 */                     /* 4 */
ViInt32   gate =     2;  /* SMU2 */
ViInt32   source =   4;  /* SMU4 */
ViInt32   bulk =     6;  /* SMU6 */

ViReal64  vd =       3;
ViReal64  vg =       3;
ViReal64  idcomp =   0.05;
ViReal64  igcomp =   0.01;
ViReal64  hold =     0;
ViReal64  delay =    0;
ViReal64  s_delay =  0;
ViReal64  pdcomp =   0;
ViReal64  pgcomp =   0;
ViInt32   nop =      11;

ViInt32   rep;
ViReal64  sc[11];
ViReal64  md[11];
ViInt32   st[11];
ViReal64  tm[11];
ViInt32   i;

ViChar    f_name[] = "C:\Agilent\data\data2.txt";
ViChar    head1[] = "Vg (V),  Id (mA), Time (sec), Status";
ViChar    msg1[] = "Saving data...";
ViChar    msg2[] = "Data save completed.";
ViChar    c = '\n';                                     /* 31 */

ret = age5270_setSwitch(vi, drain, 1);                  /* 33 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 37 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 31	Declares variables, and defines the value.
33 to 36	Enables measurement channels.
37	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
ret = age5270_resetTimestamp(vi);                                          /* 39 */
ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);

ret = age5270_setIv(vi, gate, age5270_SWP_VF_SGLLIN, 0, 0, vg, nop, hold, delay,
s_delay, igcomp, pgcomp);
check_err (vi, ret);                                                       /* 44 */

ret = age5270_setSweepSync(vi, drain, age5270_VF_MODE, 0, 0, vd, idcomp, pdcomp);
check_err (vi, ret);

ret = age5270_sweepIv(vi, drain, age5270_IM_MODE, 0, &rep, &sc[0], &md[0], &st[0],
&tm[0]);
check_err (vi, ret);                                                       /* 50 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                              /* 52 */
ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
check_err (vi, ret);                                                       /* 54 */

if ( rep != nop ) {                                                        /* 56 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }                                                                         /* 59 */

printf(" Vg (V),  Id (mA)\n");                                             /* 61 */
for (i = 0; i < nop; i++){
   printf(" %4.2f,    %9.6f \n", sc[i], md[i] * 1000);
 }                                                                         /* 64 */
```

Line	Description
39	Resets time stamp.
40 to 41	Applies voltage to device.
43	Sets the primary sweep source.
46	Sets the synchronous sweep source by using the age5270_setSweepSync function.
50	Performs staircase sweep measurement by using the age5270_sweepIv function.
52	Sets the specified port to the zero output state.
53	Disables all ports.
44, 47, 50, and 54	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
56 to 59	Stops the program execution if the number of returned data is not equal to nop.
61 to 64	Displays the measurement result data.

```
FILE *stream;                                                          /* 66 */

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, %9.6f, %8.6f, %d\n", sc[i], md[i] * 1000, tm[i],
st[i]);
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                     /* 82 */
}
```

Line	Description
66 to 82	Saves the measurement results into a file (C:\Agilent\data\data2.txt, CSV file).
83	End of the perform_meas subprogram.

Measurement Result Example

```
Vg (V),  Id (mA), Time (sec), Status
0.00,  0.004043, 0.065200, 0
0.30,  2.330500, 0.071300, 0
0.60,  4.904000, 0.073000, 0
0.90,  7.723500, 0.074600, 0
1.20, 10.753000, 0.076300, 0
1.50, 13.975000, 0.080000, 0
1.80, 17.385000, 0.081200, 0
2.10, 20.955000, 0.082800, 0
2.40, 24.660000, 0.084300, 0
2.70, 28.500000, 0.085500, 0
3.00, 32.450000, 0.087000, 0
```

Table 5-7 uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 5-6, MOSFET Id-Vg measurement).

Table 5-7 Staircase Sweep Measurement Example 3

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32   drain =     1;  /* SMU1 */                    /* 4 */
ViInt32   gate =      2;  /* SMU2 */
ViInt32   source =    4;  /* SMU4 */
ViInt32   bulk =      6;  /* SMU6 */

ViReal64  vd =        3;
ViReal64  vg =        3;
ViReal64  idcomp =    0.05;
ViReal64  igcomp =    0.01;
ViReal64  hold =      0;
ViReal64  delay =     0;
ViReal64  s_delay =   0;
ViReal64  pdcomp =    0;
ViReal64  pgcomp =    0;
ViInt32   nop =       11;

ViInt32   rep;
ViReal64  sc[11];
ViReal64  md[11];
ViInt32   st[11];
ViReal64  tm[11];
ViInt32   i;

ViChar    f_name[] = "C:\Agilent\data\data3.txt";
ViChar    head1[] = "Vg (V),  Id (mA), Time (sec), Status";
ViChar    msg1[] = "Saving data...";
ViChar    msg2[] = "Data save completed.";
ViChar    c = '\n';                                     /* 31 */

ret = age5270_setSwitch(vi, drain, 1);                  /* 33 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 37 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 31	Declares variables, and defines the value.
33 to 36	Enables measurement channels.
37	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
ret = age5270_resetTimestamp(vi);                                          /* 39 */
ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);

ret = age5270_setIv(vi, gate, age5270_SWP_VF_SGLLIN, 0, 0, vg, nop, hold, delay,
s_delay, igcomp, pgcomp);
check_err (vi, ret);                                                       /* 44 */

ret = age5270_setNthSweep(vi, 2, drain, age5270_VF_MODE, 0, 0, vd, idcomp, pdcomp);
check_err (vi, ret);

ret = age5270_msweepIv(vi, drain, age5270_IM_MODE, 0, &rep, &sc[0], &md[0], &st[0],
&tm[0]);
check_err (vi, ret);                                                       /* 50 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                              /* 52 */
ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
check_err (vi, ret);                                                       /* 54 */

if ( rep != nop ) {                                                        /* 56 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }                                                                         /* 59 */

printf(" Vg (V),  Id (mA)\n");                                             /* 61 */
for (i = 0; i < nop; i++){
   printf(" %4.2f,    %9.6f \n", sc[i], md[i] * 1000);
 }                                                                         /* 64 */
```

Line	Description
39	Resets time stamp.
40 to 41	Applies voltage to device.
43	Sets the primary sweep source.
46	Sets the synchronous sweep source by using the age5270_setNthSweep function.
50	Performs staircase sweep measurement by using the age5270_msweepIv function.
52	Sets the specified port to the zero output state.
53	Disables all ports.
44, 47, 50, and 54	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
56 to 59	Stops the program execution if the number of returned data is not equal to nop.
61 to 64	Displays the measurement result data.

```
FILE *stream;                                                          /* 66 */

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, %9.6f, %8.6f, %d\n", sc[i], md[i] * 1000, tm[i],
st[i]);
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                     /* 82 */
}
```

Line	Description
66 to 82	Saves the measurement results into a file (C:\Agilent\data\data3.txt, CSV file).
83	End of the perform_meas subprogram.

Measurement Result Example

```
Vg (V),  Id (mA), Time (sec), Status
0.00, -0.000117, 0.071900, 0
0.30,  2.337500, 0.090900, 0
0.60,  4.930500, 0.092500, 0
0.90,  7.764500, 0.094100, 0
1.20, 10.812500, 0.095800, 0
1.50, 14.050000, 0.099300, 0
1.80, 17.475000, 0.100500, 0
2.10, 21.050000, 0.102100, 0
2.40, 24.765000, 0.103600, 0
2.70, 28.600000, 0.105200, 0
3.00, 32.560000, 0.106500, 0
```

Multi Channel Sweep Measurement

Table 5-8 explains an example subprogram that performs the multi channel sweep measurement. The following subprogram performs I-V measurement and saves the measurement results (bipolar transistor Ic-Vb and Ib-Vb characteristics) into a file.

Table 5-8 **Multi Channel Sweep Measurement Example 1**

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32    emitter =   1;   /* SMU1 */                  /* 3 */
ViInt32    base =      2;   /* SMU2 */
ViInt32    collector = 4;   /* SMU4 */
ViReal64   vb1 =       0.3;
ViReal64   vb2 =       0.8;
ViReal64   vc =        3;
ViReal64   ve =        0;
ViReal64   ibcomp =    0.01;
ViReal64   iccomp =    0.1;
ViReal64   iecomp =    0.1;
ViReal64   pcomp =     0;
ViInt32    nop =       11;
ViReal64   hold =      0;
ViReal64   delay =     0;
ViReal64   s_delay =   0;
ViReal64   p_comp =    0;
ViInt32    smpl =      5;
ViInt32    mch[3];
ViInt32    mode[2];
ViReal64   range[2];
ViInt32    rep;
ViReal64   sc[11];
ViReal64   md[22];
ViInt32    st[22];
ViReal64   tm[22];
mch[0] =   collector;
mch[1] =   base;
mch[2] =   0;
mode[0] =  1;
mode[1] =  1;
range[0] = 0;
range[1] = 0;                                           /* 34 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 34	Declares variables, and defines the value.

```
ret = age5270_setSwitch(vi, emitter, 1);                                    /* 36 */
ret = age5270_setSwitch(vi, base, 1);
ret = age5270_setSwitch(vi, collector, 1);
check_err (vi, ret);                                                        /* 39 */

ret = age5270_setAdc(vi, age5270_HSPEED_ADC, age5270_INTEG_MANUAL, smpl,
age5270_FLAG_OFF);
ret = age5270_setAdcType(vi, age5270_CH_ALL, age5270_HSPEED_ADC);           /* 42 */

ret = age5270_resetTimestamp(vi);
check_err (vi, ret);                                                        /* 45 */

ret = age5270_force(vi, emitter, age5270_VF_MODE, 0, ve, iecomp, 0);        /* 47 */
ret = age5270_force(vi, collector, age5270_VF_MODE, 0, vc, iccomp, 0);
ret = age5270_setIv(vi, base, age5270_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp);
check_err (vi, ret);                                                        /* 50 */

ret = age5270_sweepMiv(vi, mch, mode, range, &rep, &sc[0], &md[0], &st[0], &tm[0]);
check_err (vi, ret);                                                        /* 53 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                               /* 55 */
ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
check_err (vi, ret);                                                        /* 57 */

if ( rep != nop ) {                                                         /* 59 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }
```

Line	Description
36 to 38	Enables measurement channels.
41 to 42	Sets the high speed ADC, and selects it for all measurement channels.
44	Resets time stamp.
47 to 49	Applies voltage to device, and sets the staircase sweep source.
52	Performs measurement by using the age5270_sweepMiv function.
55	Sets the specified port to the zero output state.
56	Disables all ports.
39, 45, 50, 53, and 57	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
59 to 62	Stops the program execution if the number of returned data is not equal to the nop value.

```
ViInt32    i;                                                          /* 64 */
ViInt32    n;
printf(" Vb (V),  Ic (mA),  Ib (mA)\n");
for (i = 0; i < nop; i++){
  printf(" %4.2f,  %11.8f,  %11.8f\n", sc[i], md[2*i] * 1000, md[2*i+1] * 1000);
 }

ViChar     f_name[] = "C:\Agilent\data\data4.txt";                     /* 71 */
ViChar     head1[] =  "Vb (V), Ic (mA), Ib (mA), hfe, Tc (sec), Tb (sec), Status_c,
Status_b";
ViChar     msg1[] =   "Saving data...";
ViChar     msg2[] =   "Data save completed.";
ViChar     c =        '\n';
FILE *stream;
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, %11.8f, %11.8f, %12.8f, %8.6f, %8.6f, %d, %d\n",
sc[i], md[2*i] * 1000, md[2*i+1] * 1000, md[2*i]/md[2*i+1], tm[2*i], tm[2*i+1],
st[2*i], st[2*i+1]);
   }
  printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
  printf( "Data file was not closed\n" );
 }
}                                                                      /* 93 */
```

Line	Description
64 to 69	Displays the measurement result data.
71 to 92	Saves the measurement results into a file (C:\Agilent\data\data4.txt, CSV file).
93	End of the perform_meas subprogram.

Measurement Result Example

```
Vb (V), Ic (mA), Ib (mA), hfe, Tc (sec), Tb (sec), Status_c, Status_b
0.30,  0.00000083, -0.00000001, -59.41935484, 0.093200, 0.127000, 0, 0
0.35,  0.00000557,  0.00000005, 123.29646018, 0.168700, 0.197800, 0, 0
0.40,  0.00003837,  0.00000032, 119.64452760, 0.286700, 0.302100, 0, 0
0.45,  0.00026580,  0.00000190, 140.15291326, 0.354400, 0.355500, 0, 0
0.50,  0.00185550,  0.00001155, 160.64935065, 0.384400, 0.389200, 0, 0
0.55,  0.01274500,  0.00007378, 172.73158501, 0.396900, 0.398000, 0, 0
0.60,  0.08796500,  0.00047225, 186.26786660, 0.405800, 0.407100, 0, 0
0.65,  0.60135000,  0.00303550, 198.10574864, 0.415600, 0.420900, 0, 0
0.70,  3.84650000,  0.01897000, 202.76752768, 0.428700, 0.429800, 0, 0
0.75, 18.79500000,  0.09735000, 193.06625578, 0.433900, 0.435000, 0, 0
0.80, 55.71000000,  0.33300000, 167.29729730, 0.437900, 0.441000, 0, 0
```

Table 5-9 uses the multi channel sweep measurement mode to perform the same measurement as the previous example (Table 5-8, Ic-Vb, Ib-Vb).

Table 5-9 Multi Channel Sweep Measurement Example 2

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32     emitter =   1;    /* SMU1 */                /* 3 */
ViInt32     base =      2;    /* SMU2 */
ViInt32     collector = 4;    /* SMU4 */
ViReal64    vb1 =       0.25;
ViReal64    vb2 =       0.75;
ViReal64    vc =        3;
ViReal64    ve =        0;
ViReal64    ibcomp =    0.01;
ViReal64    iccomp =    0.1;
ViReal64    iecomp =    0.1;
ViReal64    pcomp =     0;
ViInt32     nop =       11;
ViReal64    hold =      0;
ViReal64    delay =     0;
ViReal64    s_delay =   0;
ViReal64    p_comp =    0;
ViInt32     smpl =      5;
ViInt32     mch[3];
ViInt32     mode[2];
ViReal64    range[2];
ViInt32     rep;
ViReal64    sc[11];
ViReal64    md[22];
ViInt32     st[22];
ViReal64    tm[22];
mch[0] =    collector;
mch[1] =    base;
mch[2] =    0;
mode[0] =   1;
mode[1] =   1;
range[0] = -0.1;
range[1] = -0.0001;                                     /* 34 */

ret = age5270_setSwitch(vi, emitter, 1);                /* 36 */
ret = age5270_setSwitch(vi, base, 1);
ret = age5270_setSwitch(vi, collector, 1);
check_err (vi, ret);                                    /* 39 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 34	Declares variables, and defines the value.
36 to 39	Enables measurement channels.

```
ret = age5270_setAdc(vi, age5270_HSPEED_ADC, age5270_INTEG_MANUAL, smpl,
age5270_FLAG_OFF);                                                    /* 41 */
ret = age5270_setAdcType(vi, age5270_CH_ALL, age5270_HSPEED_ADC);

ret = age5270_resetTimestamp(vi);
check_err (vi, ret);                                                  /* 45 */

ret = age5270_force(vi, emitter, age5270_VF_MODE, 0, ve, iecomp, 0);  /* 47 */
ret = age5270_force(vi, collector, age5270_VF_MODE, 0, vc, iccomp, 0);
ret = age5270_setIv(vi, base, age5270_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp);
check_err (vi, ret);                                                  /* 50 */

ret = age5270_msweepMiv(vi, mch, mode, range, &rep, &sc[0], &md[0], &st[0],
&tm[0]);
check_err (vi, ret);                                                  /* 53 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                         /* 55 */
ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
check_err (vi, ret);                                                  /* 57 */

if ( rep != nop ) {                                                   /* 59 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }

ViInt32    i;                                                         /* 64 */
ViInt32    n;
printf(" Vb (V),  Ic (mA),  Ib (mA)\n");
for (i = 0; i < nop; i++){
  printf(" %4.2f,  %9.6f,  %9.6f\n", sc[i], md[2*i] * 1000, md[2*i+1] * 1000);
 }                                                                    /* 69 */
```

Line	Description
41 to 42	Sets the high speed ADC, and selects it for all measurement channels.
44	Resets time stamp.
47 to 49	Applies voltage to device, and sets the staircase sweep source.
52	Performs measurement by using the age5270_msweepMiv function.
55	Sets the specified port to the zero output state.
56	Disables all ports.
39, 45, 50, 53, and 57	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
59 to 62	Stops the program execution if the number of returned data is not equal to nop.
64 to 69	Displays the measurement result data.

```
ViChar     f_name[] = "C:\Agilent\data\data5.txt";                          /* 71 */
ViChar     head1[] =  "Vb (V), Ic (mA), Ib (mA), hfe, Tc (sec), Tb (sec), Status_c,
Status_b";
ViChar     msg1[] =   "Saving data...";
ViChar     msg2[] =   "Data save completed.";
ViChar     c =        '\n';
FILE *stream;
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, %9.6f, %9.6f, %12.6f, %8.6f, %8.6f, %d, %d\n", sc[i],
md[2*i] * 1000, md[2*i+1] * 1000, md[2*i]/md[2*i+1], tm[2*i], tm[2*i+1], st[2*i],
st[2*i+1]);
   }
  printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
  printf( "Data file was not closed\n" );
 }
}                                                                          /* 93 */
```

Line	Description
71 to 92	Saves the measurement results into a file (C:\Agilent\data\data5.txt, CSV file).
93	End of the perform_meas subprogram.

Measurement Result Example

```
Vb (V), Ic (mA), Ib (mA), hfe, Tc (sec), Tb (sec), Status_c,
Status_b
0.25, -0.005000, -0.000005,  1000.000000, 0.058700, 0.058700, 0, 0
0.30, -0.005000, -0.000005,  1000.000000, 0.061000, 0.061000, 0, 0
0.35, -0.005000, -0.000015,   333.333333, 0.063000, 0.063000, 0, 0
0.40,  0.000000, -0.000005,     0.000000, 0.065000, 0.065000, 0, 0
0.45, -0.005000,  0.000005, -1000.000000, 0.067000, 0.067000, 0, 0
0.50,  0.000000,  0.000005,     0.000000, 0.068900, 0.068900, 0, 0
0.55,  0.010000,  0.000085,   117.647059, 0.070500, 0.070500, 0, 0
0.60,  0.085000,  0.000475,   178.947368, 0.072400, 0.072400, 0, 0
0.65,  0.595000,  0.003035,   196.046129, 0.074400, 0.074400, 0, 0
0.70,  3.825000,  0.018935,   202.006866, 0.076400, 0.076400, 0, 0
0.75, 18.740000,  0.096725,   193.745154, 0.078400, 0.078400, 0, 0
```

Pulsed Sweep Measurement

Table 5-10 explains an example subprogram that performs the pulsed sweep measurement and saves the measurement results (bipolar transistor Ic-Vc characteristics) into a file.

Table 5-10 Pulsed Sweep Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32   emitter =    1;   /* SMU1 */
ViInt32   base =       2;   /* SMU2 */
ViInt32   collector =  4;   /* SMU4 */
ViReal64  vc =         3;
ViReal64  ib =         150E-6;
ViReal64  iccomp =     0.05;
ViReal64  vbcomp =     5;
ViReal64  hold =       0.1;
ViReal64  width =      0.001;
ViReal64  period =     0.01;
ViInt32   nop1 =       11;
ViInt32   nop2 =       3;
ViInt32   rep;
ViReal64  sc[33];
ViReal64  md[33];
ViInt32   st[33];
ViReal64  tm[33];
ViReal64  dib[3];
ViInt32   i =          0;
ViInt32   j;
ViInt32   n;
ViInt32   smpl =       5;
ViChar    f_name[] =   "C:\Agilent\data\data6.txt";
ViChar    head1[] =    "Ib (uA), Vc (V), Ic (mA), Time (sec), St
atus";
ViChar    msg1[] =     "Saving data...";
ViChar    msg2[] =     "Data save completed.";
ViChar    c = '\n';

ret = age5270_setSwitch(vi, emitter, 1);              /* 31 */
ret = age5270_setSwitch(vi, base, 1);
ret = age5270_setSwitch(vi, collector, 1);
check_err (vi, ret);                                  /* 34 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
3 to 29	Declares variables, and defines the value.
31 to 33	Enables measurement channels.

```
ret = age5270_setAdc(vi, age5270_HSPEED_ADC, age5270_INTEG_MANUAL, smpl, age5270_F
LAG_OFF);                                                                    /* 36 */
ret = age5270_setAdcType(vi, age5270_CH_ALL, age5270_HSPEED_ADC);
ret = age5270_resetTimestamp(vi);
ret = age5270_force(vi, emitter, age5270_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                                  /* 41 */
   dib[j] = (j + 1) * ib / nop2;
   ret = age5270_force(vi, base, age5270_IF_MODE, 0, dib[j], vbcomp, 0);
   ret = age5270_setPiv(vi, collector, age5270_SWP_VF_SGLLIN, 0, base, 0, vc, nop1,
hold, width, period, iccomp);
   check_err (vi, ret);

   ret = age5270_sweepPiv(vi, collector, age5270_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[i], &tm[i]);
   check_err (vi, ret);

   if ( rep = nop1 ) {
      i = i + nop1;
    }
   else {
      printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep,
nop1);
      ret = age5270_zeroOutput(vi, age5270_CH_ALL);
      ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                           /* 60 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                                /* 62 */
check_err (vi, ret);
```

Line	Description
36 to 37	Sets the high speed ADC, and selects it for all measurement channels.
38	Resets time stamp.
39	Applies voltage to device.
41 to 60	Applies dc current and pulsed sweep voltage, and performs pulsed sweep measurement. After that, disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
62	Sets the specified port to the zero output state.
34, 45, 48, 57, and 63	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
printf(" Ib (uA),  Vc (V),  Ic (mA)\n");                                    /* 64 */

for (j = 0; j < nop2; j++){
   n = j * nop1;
   for (i = n; i < n + nop1; i++){
      printf(" %5.1f,    %4.2f,    %9.6f \n", dib[j] * 1E6, sc[i], md[i] * 1000);
    }
 }                                                                          /* 71 */

FILE *stream;                                                               /* 73 */

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 (j = 0; j < nop2; j++){
      n = j * nop1;
      for (i = n; i < n + nop1; i++){
         fprintf( stream, "%5.1f, %4.2f, %9.6f, %8.6f, %d\n", dib[j] * 1E6, sc[i],
md[i] * 1000, tm[i], st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                          /* 92 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);                             /* 94 */
check_err (vi, ret);

}
```

Line	Description
64 to 71	Displays the measurement result data.
73 to 92	Saves the measurement results into a file (C:\Agilent\data\data6.txt, CSV file).
94	Disables all ports.
95	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
97	End of the perform_meas subprogram.

Measurement Result Example

```
Ib (uA), Vc (V), Ic (mA), Time (sec), Status
 50.0, 0.00, -0.050000, 0.152900, 0
 50.0, 0.30,  9.015000, 0.162900, 0
 50.0, 0.60,  9.760000, 0.172900, 0
 50.0, 0.90,  9.825000, 0.182900, 0
 50.0, 1.20,  9.840000, 0.192900, 0
 50.0, 1.50,  9.875000, 0.202900, 0
 50.0, 1.80,  9.905000, 0.212900, 0
 50.0, 2.10,  9.950000, 0.222900, 0
 50.0, 2.40,  9.935000, 0.232900, 0
 50.0, 2.70,  9.970000, 0.242900, 0
 50.0, 3.00, 10.010000, 0.252900, 0
100.0, 0.00, -0.095000, 0.402900, 0
100.0, 0.30, 15.765000, 0.412900, 0
100.0, 0.60, 18.245000, 0.422900, 0
100.0, 0.90, 18.910000, 0.432900, 0
100.0, 1.20, 19.030000, 0.442900, 0
100.0, 1.50, 19.105000, 0.452900, 0
100.0, 1.80, 19.200000, 0.462900, 0
100.0, 2.10, 19.250000, 0.472900, 0
100.0, 2.40, 19.310000, 0.482900, 0
100.0, 2.70, 19.385000, 0.492900, 0
100.0, 3.00, 19.420000, 0.502900, 0
150.0, 0.00, -0.145000, 0.652900, 0
150.0, 0.30, 21.140000, 0.662900, 0
150.0, 0.60, 24.710000, 0.672900, 0
150.0, 0.90, 26.660000, 0.682900, 0
150.0, 1.20, 27.505000, 0.692900, 0
150.0, 1.50, 27.800000, 0.702900, 0
150.0, 1.80, 27.935000, 0.712900, 0
150.0, 2.10, 28.050000, 0.722900, 0
150.0, 2.40, 28.205000, 0.732900, 0
150.0, 2.70, 28.285000, 0.742900, 0
150.0, 3.00, 28.330000, 0.752900, 0
```

Staircase Sweep with Pulsed Bias Measurement

Table 5-11 explains an example subprogram that performs the staircase sweep with pulsed bias measurement and saves the measurement results (MOSFET Id-Vd characteristics) into a file.

Table 5-11 Staircase Sweep with Pulsed Bias Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32   drain =     1;  /* SMU1 */                    /* 4 */
ViInt32   gate =      2;  /* SMU2 */
ViInt32   source =    4;  /* SMU4 */
ViInt32   bulk =      6;  /* SMU6 */
ViReal64  vd =        3;
ViReal64  vg =        3;
ViReal64  idcomp =    0.05;
ViReal64  igcomp =    0.01;
ViReal64  hold =      0;
ViReal64  delay =     0;
ViReal64  s_delay =   0;
ViReal64  p_comp =    0;
ViReal64  width =     0.001;
ViReal64  period =    0.01;
ViReal64  p_hold =    0.1;
ViInt32   nop1 =      11;
ViInt32   nop2 =      3;
ViInt32   i =         0;
ViInt32   j;
ViInt32   n;

ViInt32   rep;
ViReal64  sc[33];
ViReal64  md[33];
ViInt32   st[33];
ViReal64  tm[33];
ViReal64  dvg[3];

ViChar    f_name[] = "C:\Agilent\data\data7.txt";
ViChar    head1[] =  "Vg (V), Vd (V), Id (mA), Time (sec), Stat
us";
ViChar    msg1[] =   "Saving data...";
ViChar    msg2[] =   "Data save completed.";
ViChar    c =        '\n';                              /* 36 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 36	Declares variables, and defines the value.

```
ret = age5270_setSwitch(vi, drain, 1);                                    /* 38 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                                      /* 42 */

ret = age5270_resetTimestamp(vi);                                         /* 44 */
ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                               /* 48 */
   dvg[j] = (j + 1) * vg / nop2;
   ret = age5270_setPbias(vi, gate, age5270_VF_MODE, 0, 0, dvg[j], width, period,
p_hold, igcomp);
   ret = age5270_setIv(vi, drain, age5270_SWP_VF_SGLLIN, 0, 0, vd, nop1, hold,
delay, s_delay, idcomp, p_comp);
   check_err (vi, ret);

   ret = age5270_sweepPbias(vi, drain, age5270_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[i], &tm[i]);
   check_err (vi, ret);

   if ( rep = nop1 ) {
      i = i + nop1;
    }
   else {
      printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep,
nop1);
      ret = age5270_zeroOutput(vi, age5270_CH_ALL);
      ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                        /* 67 */

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                             /* 69 */
check_err (vi, ret);
```

Line	Description
38 to 41	Enables measurement channels.
44	Resets time stamp.
45 to 46	Applies voltage to device.
48 to 67	Applies pulsed voltage and sweep voltage, and performs staircase sweep measurement. After that, disables all ports and stops the program execution if the number of returned data is not equal to the nop1 value.
69	Sets the specified port to the zero output state.
42 and 70	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
printf(" Vg (V),  Vd (V),  Id (mA)\n");                                    /* 72 */

for (j = 0; j < nop2; j++){
   n = j * nop1;
   for (i = n; i < n + nop1; i++){
      printf(" %4.2f,    %4.2f,    %9.6f \n", dvg[j], sc[i], md[i] * 1000);
    }
 }                                                                         /* 79 */

FILE *stream;                                                              /* 81 */

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 (j = 0; j < nop2; j++){
      n = j * nop1;
      for (i = n; i < n + nop1; i++){
         fprintf( stream, "%4.2f, %4.2f, %9.6f, %8.6f, %d\n", dvg[j], sc[i], md[i]
* 1000, tm[i], st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                         /* 100 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);                            /* 102 */
check_err (vi, ret);

}
```

Line	Description
72 to 79	Displays the measurement result data.
81 to 100	Saves the measurement results into a file (C:\Agilent\data\data7.txtxt, CSV file).
102	Disables all ports.
103	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
105	End of the perform_meas subprogram.

Measurement Result Example

```
Vg (V), Vd (V), Id (mA), Time (sec), Status
1.00, 0.00,  0.005000, 0.166900, 0
1.00, 0.30,  3.170000, 0.176900, 0
1.00, 0.60,  5.835000, 0.186900, 0
1.00, 0.90,  8.040000, 0.196900, 0
1.00, 1.20,  9.905000, 0.206900, 0
1.00, 1.50, 11.530000, 0.216900, 0
1.00, 1.80, 12.965000, 0.226900, 0
1.00, 2.10, 14.270000, 0.236900, 0
1.00, 2.40, 15.425000, 0.246900, 0
1.00, 2.70, 16.495000, 0.256900, 0
1.00, 3.00, 17.460000, 0.266900, 0
2.00, 0.00,  0.005000, 0.417900, 0
2.00, 0.30,  4.165000, 0.427900, 0
2.00, 0.60,  7.875000, 0.437900, 0
2.00, 0.90, 11.135000, 0.447900, 0
2.00, 1.20, 13.945000, 0.457900, 0
2.00, 1.50, 16.370000, 0.467900, 0
2.00, 1.80, 18.470000, 0.477900, 0
2.00, 2.10, 20.320000, 0.487900, 0
2.00, 2.40, 21.950000, 0.497900, 0
2.00, 2.70, 23.430000, 0.507900, 0
2.00, 3.00, 24.780000, 0.517900, 0
3.00, 0.00,  0.000000, 0.670500, 0
3.00, 0.30,  5.035000, 0.680500, 0
3.00, 0.60,  9.650000, 0.690500, 0
3.00, 0.90, 13.835000, 0.700500, 0
3.00, 1.20, 17.575000, 0.710500, 0
3.00, 1.50, 20.895000, 0.720500, 0
3.00, 1.80, 23.810000, 0.730500, 0
3.00, 2.10, 26.355000, 0.740500, 0
3.00, 2.40, 28.615000, 0.750500, 0
3.00, 2.70, 30.615000, 0.760500, 0
3.00, 3.00, 32.410000, 0.770500, 0
```

Breakdown Voltage Measurement

Table 5-12 explains an example subprogram that performs the quasi pulsed spot measurement and displays the measurement result data (bipolar transistor breakdown voltage).

Table 5-12 Breakdown Voltage Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32    emitter   = 1;   /* SMU1 */
/*ViInt32  base;              open */
ViInt32    collector = 4;   /* SMU4 */
ViReal64   start =     0;
ViReal64   vc =        100; /* intlk cable must be connected */
ViReal64   iccomp =    0.005;
ViReal64   hold =      0;
ViReal64   delay =     0;
ViReal64   meas;
ViInt32    status;                                      /* 13 */

ret = age5270_setSwitch(vi, emitter, 1);                /* 15 */
ret = age5270_setSwitch(vi, collector, 1);
check_err (vi, ret);                                    /* 17 */

ret = age5270_force(vi, emitter, age5270_VF_MODE, 0, 0, 0.1, 0);
check_err (vi, ret);                                    /* 20 */

ret = age5270_setBdv(vi, collector, 0, start, vc, iccomp, hold,
delay);
check_err (vi, ret);                                    /* 23 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 13	Declares variables, and defines the value.
15 to 16	Enables measurement channels.
19	Applies voltage to device.
22	Sets the quasi pulsed voltage source.
17, 20, and 23	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
ret = age5270_measureBdv(vi, age5270_SHORT_INTERVAL, &meas,
&status);                                                    /* 25 */
check_err (vi, ret);

ret = age5270_zeroOutput(vi, age5270_CH_ALL);                /* 28 */
check_err (vi, ret);

if (status = 8){                                             /* 31 */
   printf("Vbd = %9.6f V \n", meas);
 }
else {
   printf("Error occurred during measurement.\n");
   printf("Status code = %d \n", status);
 }                                                           /* 37 */

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);              /* 39 */
check_err (vi, ret);
}
```

Line	Description
25	Performs quasi pulsed spot measurement. Breakdown voltage will be defined as the voltage that occurs the current compliance status at the device terminal where the measurement channel is connected.
28	Sets the specified port to the zero output state.
31 to 37	Displays the measurement result data if the status is normal (8), or displays error message if the status is abnormal.
39	Disables all ports.
26, 29, and 40	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
41	End of the perform_meas subprogram.

Measurement Result Example

```
Vbd = 56.245000 V
```

Leakage Current Measurement

Table 5-13 explains an example subprogram that performs the quasi pulsed spot measurement and displays the measurement result data (MOSFET drain current).

Table 5-13 Leakage Current Measurement Example

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32    drain =   1;  /* SMU1, drain */
ViInt32    gate =    2;  /* SMU2, gate */
ViInt32    source =  4;  /* SMU4, source */
ViInt32    bulk =    6;  /* SMU6, bulk */
ViReal64   vd =      5;
ViReal64   vg =      0;
ViReal64   idcomp =  0.05;
ViReal64   igcomp =  0.01;
ViReal64   start =   -5;
ViReal64   hold =    0.1;
ViReal64   delay =   0.001;
ViReal64   meas;
ViInt32    status;                                      /* 16 */

ret = age5270_setSwitch(vi, drain, 1);                  /* 18 */
ret = age5270_setSwitch(vi, gate, 1);
ret = age5270_setSwitch(vi, source, 1);
ret = age5270_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 22 */

ret = age5270_force(vi, bulk, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, source, age5270_VF_MODE, 0, 0, 0.1, 0);
ret = age5270_force(vi, gate, age5270_VF_MODE, 0, vg, igcomp,
0);
check_err (vi, ret);                                    /* 27 */
```

Line	Description
1	Beginning of the perform_meas subprogram.
4 to 16	Declares variables, and defines the value.
18 to 21	Enables measurement channels.
24 to 26	Applies voltage to device.
22 and 27	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.

```
ret = age5270_setIleak(vi, drain, 0, vd, idcomp, start, hold,
delay);                                                    /* 29 */
check_err (vi, ret);

ret = age5270_measureIleak(vi, drain, age5270_SHORT_INTERVAL,
&meas, &status);                                           /* 32 */
check_err (vi, ret);

ret = age5270_zeroOutput(vi, age5270_CH_ALL);              /* 35 */
check_err (vi, ret);

printf("Id = %9.6f mA\n", meas * 1000);                    /* 38 */
printf("Vd = %5.2f to %4.2f V\n", start, vd);
printf("Vg = %4.2f V\n", vg);

ret = age5270_setSwitch(vi, age5270_CH_ALL, 0);            /* 42 */
check_err (vi, ret);
}
```

Line	Description
29	Sets the quasi pulsed voltage source.
32	Performs quasi pulsed spot measurement. Leakage current will be defined as the current when the target voltage (vd) is applied to device terminal where the source channel is connected.
35	Sets the specified port to the zero output state.
38 to 40	Displays the measurement result data.
42	Disables all ports.
30, 33, 36, and 43	Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line.
44	End of the perform_meas subprogram.

Measurement Result Example

```
Id = 12.240000 mA
Vd = -5.00 to 5.00 V
Vg = 0.00 V
```