# VXI*plug&play* Driver User's Guide

## Agilent 4155C Semiconductor Parameter Analyzer
## Agilent 4156C Precision Semiconductor Parameter Analyzer

Agilent Technologies

**04156-90083**

**March 2004**

**Edition 3**

# Legal Notice

The information contained in this document is subject to change without notice.



- **Product Warranty**

  Agilent Technologies warrant Agilent Technologies hardware, accessories and supplies against defects in materials and workmanship for the period of one year from the warranty start date specified below. If Agilent Technologies receive notice of such defects during the warranty period, Agilent Technologies will, at its option, either repair or replace products which prove to be defective. Replacement products may be either new or like-new.

  Warranty service of this product will be performed at Agilent Technologies. Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to Buyer. However, Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

  Agilent Technologies do not warrant that the operation of Agilent Technologies products will be uninterrupted or error free. If Agilent is unable, within a reasonable time, to repair or replace any product to a condition as warranted, customer will be entitled to a refund of the purchase price upon prompt return of the product.

  The Agilent Technologies products may contain remanufactured parts equivalent to new in performance or may have been subject to incidental use.

  The warranty period begins on the date of delivery or on the date of installation if installed by Agilent Technologies. If customer schedules or delays Agilent Technologies installation more than 30 days after delivery, warranty begins on the 31st day from delivery.

  Warranty does not apply to defects resulting from (a) improper or inadequate maintenance or calibration, (b) software, interfacing, parts or supplies not supplied by Agilent Technologies, (c) unauthorized modification or misuse, (d) operation outside of the published environmental specifications for the product, or (e) improper site preparation or maintenance.

To the extent allowed by local law, the above warranties are exclusive and no other warranty or condition, whether written or oral, is expressed or implied and Agilent Technologies specifically disclaim any implied warranties or conditions of merchantability, satisfactory quality, and fitness for a particular purpose.

Agilent Technologies will be liable for damage to tangible property per incident up to the greater of $300,000 or the actual amount paid for the product that is the subject of the claim, and for damages for bodily injury or death, to the extent that all such damages are determined by a court of competent jurisdiction to have been directly caused by a defective Agilent Technologies product.

To the extent allowed by local law, the remedies in this warranty statement are customer's sole and exclusive remedies. Expect as indicated above, in no event will Agilent Technologies or its suppliers be liable for loss of date or for direct, special, incidental, consequential (including lost profit or date), or other damage, whether based in contract, tort, or otherwise.

For consumer transactions in Australia and New Zealand: the warranty terms contained in this statement, except to the extent lawfully permitted, do not exclude, restrict or modify and are in addition to the mandatory statutory rights applicable to the sale of this product to you.

- **Assistance**

  Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

  For any assistance, contact your nearest Agilent Technologies Sales Office.

- **Certification**

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

# In This Manual

This manual provides information about the 4155/4156 VXI*plug&play* driver. This manual also explains two sample application programs using the Agilent VEE and the driver.

- Installation

  This chapter describes system requirements to use the 4155/4156 VXI*plug&play* driver, and how to install the driver.

- Driver Functions

  This chapter lists the all driver functions for the Agilent 4155/4156 and the Agilent  E5250A Low Leakage Switch Mainframe.

- Programming Examples for Visual Basic Users

  This chapter describes how to create measurement program using Microsoft Visual Basic, and provides programming examples.

- Programming Examples for Visual Basic .NET Users

  This chapter describes how to create measurement program using Microsoft Visual Basic .NET, and provides programming examples.

- Programming Examples for C++ Users

  This chapter describes how to create measurement program using Microsoft Visual C++, and provides programming examples.

- Programming Examples for VEE Users

  This chapter describes how to create measurement program using Agilent VEE, and provides programming examples.

- Sample Application Programs

  This chapter provides how to install, execute, and modify the sample application programs stored in the Agilent VEE Sample Program Disk furnished with the 4155/4156.

# Printing History

Edition 1: January 2001

Edition 2: August 2003

Edition 3: March 2004

# Contents

# Contents

# Contents

# Contents

# Contents

# Contents

# Contents

# Contents

# 1 Installation

This chapter explains the environment requirements and installation of the VXI*plug&play* driver for Agilent 4155/4156.

- “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, Windows 98, or Windows 95. It must be supported by the application development environment.

- Application Development Environment (or 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 flexible disk drive (FDD) is required to install the VXI*plug&play* driver. The CD-ROM drive is required to install the software needed to use driver.

- Minimum disk space

  2 MB for 4155/4156 driver

  1 MB for E5250A driver

**NOTE** If you use the sample application programs, stored in the VEE Sample Program Disk furnished with the 4155/4156, VEE software must be version 4.0 or later. See Chapter 7. Also, if you use the Cascade Microtech Summit series semi-auto prober, confirm the operating system supported by the prober control software (PCS) supplied from Cascade Microtech, Inc.

**NOTE** The E5250A Plug&Play Driver Disk stores the VXI*plug&play* driver for Agilent E5250A. This driver is required to use the sample application programs.

# Installing VXI*plug&play* Driver

The installation flow is shown below. If you have already installed the GPIB interface card, VISA I/O library, and programming software on your PC, skip steps 1 through 4.

1. Install the GPIB interface card into your PC.

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

2. Install VISA I/O library.

   Follow the instructions in the I/O library's setup program.

3. Configure and check the GPIB interface.

   See the I/O library manual.

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 VXI*plug&play* driver as shown below.

   a. Insert the 4155/4156 Plug&Play Driver Disk into FDD of your PC.

   b. Execute SETUP.EXE stored on the diskette. The program automatically installs the following files in the folder \Vxipnp\Win*xx*\Hp4156b.

      - hp4156b.bas
      - hp4156b.c
      - hp4156b.def
      - hp4156b.fp
      - hp4156b.GID
      - hp4156b.h
      - hp4156b.hlp
      - readme.txt
      - DelsL1.isu

7. If you are also installing the driver for the E5250A, do the following.

   a. Insert the E5250A Plug&Play Driver Disk into FDD of your PC.

   b. Execute SETUP.EXE stored on the diskette. The program automatically installs the folloiwng files in the folder \Vxipnp\Win*xx*\Hpe5250a.

      - hpe5250a.bas
      - hpe5250a.c
      - hpe5250a.def
      - hpe5250a.fp
      - hpe5250a.GID
      - hpe5250a.h
      - hpe5250a.hlp
      - readme.txt
      - DelsL1.isu

**NOTE** Win*xx* will be Winnt for Windows XP, Windows 2000, and Windows NT, or Win95 for others.

# 2 Driver Function Reference

This section explains all the driver functions available for Agilent 41555/4156 and Agilent E5250A.

- “4155/4156 Driver Functions”
- “E5250A Driver Functions”

**NOTE**

For additional information on each function. refer to the on-line help for the VXI*plug&play* drivers, or open the hp4156b.hlp or hpe5250a.hlp file in the folder the driver is installed. See “Installing VXIplug&play Driver” on page 1-4.

# 4155/4156 Driver Functions

This section provides the reference information of the Agilent 4155/4156 VXI*plug&play* driver functions.

## Function List

Table 2-1 lists all the functions for the 4155/4156. You will see a brief description of the functions in the table.

For the description, syntax and parameters of the function, refer to the reference section following this table. The driver functions in the reference section will appear in the alphabetical order.

**Table 2-1 4155/4156 Driver Function List**

| Category | Function | Description |
|---|---|---|
| Miscellaneous | hp4156b_init | Initializes the 4155/4156. |
| | hp4156b_close | Closes the connection with the 4155/4156. |
| | hp4156b_reset | Executes the 4155/4156 reset. |
| | hp4156b_self_test | Executes the 4155/4156 self-test. |
| | hp4156b_error_query | Queries the 4155/4156 for error code/message. |
| | hp4156b_error_message | Queries for the driver errors. |
| | hp4156b_revision_query | Queries for the 4155/4156 firmware/driver revisions. |
| | hp4156b_timeOut | Sets the timeout. |
| | hp4156b_timeOut_Q | Queries for the timeout setting. |
| | hp4156b_errorQueryDetect | Sets the automatic error checking. |
| | hp4156b_errorQueryDetect_Q | Queries for the automatic error checking setting. |
| | hp4156b_dcl | Sends the Device Clear. |
| | hp4156b_esr_Q | Queries the ESR status. |
| | hp4156b_readStatusByte_Q | Reads the 4155/4156 status byte. |
| | hp4156b_opc_Q | Checks the 4155/4156 operation completion status. |

| Category | Function | Description |
|---|---|---|
| Primitive Measurement Functions | hp4156b_startMeasure | Starts a measurement. |
| | hp4156b_readData | Reads a measurement result. |
| | hp4156b_stopMode | Sets the measurement completion mode. |
| | hp4156b_abortMeasure | Aborts output or measurement. |
| Calibration | hp4156b_autoCal | Sets the auto calibration mode |
| | hp4156b_execCal | Executes the 4155/4156 calibration |
| Zero Offset Cancel | hp4156b_offsetCancel | Sets the zero offset cancel. |
| | hp4156b_execOffsetCancel | Executes the zero offset cancel. |
| Measurement Unit Setup | hp4156b_setSwitch | Sets the output switch. |
| | hp4156b_setFilter | Sets the output filter. |
| | hp4156b_setInteg | Sets the integration time. |
| | hp4156b_setVm | Sets the VMU measurement mode. |
| | hp4156b_setPguR | Sets the PGU output impedance. |
| Source Setup | hp4156b_force | Applies a dc current or voltage. |
| | hp4156b_forcePulse | Applies a pulse by using PGU. |
| | hp4156b_zeroOutput | Disables output. |
| | hp4156b_recoverOutput | Recovers output. |
| | hp4156b_setIv | Sets the sweep source. |
| | hp4156b_setPbias | Sets the pulsed bias source. |
| | hp4156b_setPiv | Sets the pulsed sweep source. |
| | hp4156b_setSweepSync | Sets the synchronous sweep source. |
| Measurement Execution | hp4156b_spotMeas | Executes a high speed spot measurement. |
| | hp4156b_measureM | Executes a multi-channel spot measurement. |
| | hp4156b_sweepIv | Executes a one channel sweep measurement. |
| | hp4156b_sweepMiv | Executes a multi-channel sweep measurement. |
| | hp4156b_measureP | Executes a pulsed spot measurement. |
| | hp4156b_sweepPiv | Executes a pulsed sweep measurement. |
| | hp4156b_sweepPbias | Executes a sweep measurement with pulsed bias. |

| Category | Function | Description |
|---|---|---|
| Sampling Measurements | hp4156b_setSample | Sets the timing parameters. |
| | hp4156b_addSampleSynclv | Sets the dc source. |
| | hp4156b_addSampleSyncPulse | Sets the pulse source. |
| | hp4156b_sample | Executes a sampling measurement. |
| | hp4156b_clearSampleSync | Clears the source setup. |
| Stress Force | hp4156b_setStress | Sets the timing parameters. |
| | hp4156b_addStressSyncIv | Sets the dc stress source. |
| | hp4156b_addStressSyncPulse | Sets the pulse stress source. |
| | hp4156b_stress | Forces stress. |
| | hp4156b_clearStressSync | Clears the source setup. |
| Passthrough Functions | hp4156b_cmd | Sends a command. |
| | h4156b_cmdInt | Sends a command with an integer parameter. |
| | hp4156b_cmdReal | Sends a command with a real parameter. |
| | hp4156b_cmdData_Q | Sends a command to read any data. |
| | hp4156b_cmdString_Q | Sends a command to read string response. |
| | hp4156b_cmdInt16_Q | Sends a command to read 16 bit integer response. |
| | hp4156b_cmdInt16Arr_Q | Sends a command to read 16 bit integer array response. |
| | hp4156b_cmdInt32_Q | Sends a command to read 32 bit integer response. |
| | hp4156b_cmdInt32Arr_Q | Sends a command to read 32 bit integer array response. |
| | hp4156b_cmdReal64_Q | Sends a command to read 64 bit real response. |
| | hp4156b_cmdReal64Arr_Q | Sends a command to read 64 bit real array response. |

# Parameters

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

- “range value and ranging mode”
- “VSU output voltage, resolution, and compliance”
- “SMU output voltage, resolution, and compliance by range”
- “SMU/VMU measurement voltage and resolution by range”
- “SMU output current, resolution, and compliance by range”
- “SMU measurement current and resolution by range”
- “PGU output voltage, resolution, and compliance by range”
- “PGU pulse period, pulse width, and delay time”
- “PGU leading transition time and trailing transition time”

**NOTE**

Macros

Some functions can use macros to set the parameter values. For details of functions and macros, refer to the help file (hp4156b.hlp) in the directory that the driver is installed.

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

| Voltage or current | Available *range* values [a] | Ranging mode used for output/measurement |
|---|---|---|
| both | *range* = 0 | Auto ranging [b] |
| voltage | 0 < *range* ≤ 0.2 V (for VMU in differential mode) | 0.2 V limited auto ranging [c] |
| | 0 < *range* ≤ 2 V | 2 V limited auto ranging [c] |
| | 2 V < *range* ≤ 20 V | 20 V limited auto ranging [c] |
| | 20 V < *range* ≤ 40 V | 40 V limited auto ranging [c] |
| | 40 V < *range* ≤ 100 V | 100 V limited auto ranging [c] |
| | 100 V < *range* ≤ 200 V (for HPSMU) | 200 V limited auto ranging [c] |

| Voltage or current | Available *range* values [a] | Ranging mode used for output/measurement |
|---|---|---|
| current | 0 < *range* ≤ 10 pA (for HRSMU) | 10 pA limited auto ranging [c] |
| | 10 pA < *range* ≤ 100 pA (for HRSMU) | 100 pA limited auto ranging [c] |
| | 100 pA < *range* ≤ 1 nA | 1 nA limited auto ranging [c] |
| | 1 nA < *range* ≤ 10 nA | 10 nA limited auto ranging [c] |
| | 10 nA < *range* ≤ 100 nA | 100 nA limited auto ranging [c] |
| | 100 nA < *range* ≤ 1 μA | 1 μA limited auto ranging [c] |
| | 1 μA < *range* ≤ 10 μA | 10 μA limited auto ranging [c] |
| | 10 μA < *range* ≤ 100 μA | 100 μA limited auto ranging [c] |
| | 100 μA < *range* ≤ 1 mA | 1 mA limited auto ranging [c] |
| | 1 mA < *range* ≤ 10 mA | 10 mA limited auto ranging [c] |
| | 10 mA < *range* ≤ 100 mA | 100 mA limited auto ranging [c] |
| | 100 mA < *range* ≤ 1 A (for HPSMU) | 1 A limited auto ranging [c] |

a. Negative *range* values are available for the functions used to start or execute measurements. The negative values set the ranging mode to the fix, not the limited auto.
b. SMU uses the optimum range to force/measure voltage or current.
c. SMU uses the optimum range to force/measure voltage or current. Then, the SMU never uses the range less than the specified range.

**Table 2-3 VSU output voltage, resolution, and compliance**

| Output range (actually used) | Output voltage [a] | Setting resolution | *comp* value [b] |
|---|---|---|---|
| 20 V | 0 ≤ \|V\| ≤ 20 V | 1 mV | ±100 mA |

a. Parameter name may be *base*, *bias*, *peak*, *stress*, *value*, *start*, *stop*, and so on.
b. Set any value (e.g. 0.1) though the compliance is always 100 mA.

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

| Output range (actually used) | Output voltage [a] | Setting resolution | Maximum *comp* value [b] | |
|---|---|---|---|---|
| | | | HRSMU/ MPSMU | HPSMU |
| 2 V | 0 ≤ \|V\| ≤ 2 V | 100 µV | ±100 mA | ±1000 mA |
| 20 V | 0 ≤ \|V\| ≤ 20 V | 1 mV | ±100 mA | ±1000 mA |
| 40 V | 0 ≤ \|V\| ≤ 20 V | 2 mV | ±100 mA | ±1000 mA |
| | 20 V < \|V\| ≤ 40 V | | ±50 mA | ±500 mA |
| 100 V | 0 ≤ \|V\| ≤ 20 V | 5 mV | ±100 mA | ±1000 mA |
| | 20 V < \|V\| ≤ 40 V | | ±50 mA | ±500 mA |
| | 40 V < \|V\| ≤ 100 V | | ±20 mA | ±125 mA |
| 200 V | 0 ≤ \|V\| ≤ 20 V | 10 mV | - | ±1000 mA |
| | 20 V < \|V\| ≤ 40 V | | - | ±500 mA |
| | 40 V < \|V\| ≤ 100 V | | - | ±125 mA |
| | 100 V < \|V\| ≤ 200 V | | - | ±50 mA |

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

**Table 2-5 SMU/VMU measurement voltage and resolution by range**

| Measurement range (actually used) | Measurement voltage | Measurement resolution | | | |
|---|---|---|---|---|---|
| | | Integration time | | | High speed [a] sampling measurement |
| | | 1PLC or longer | 640 µs to 1.92 ms | 80 µs to 560 µs | |
| 0.2 V (for VMU in differential mode) | 0 ≤ \|V\| ≤ 0.2 V | 0.2 µV | 2 µV | 20 µV | 200 µV |
| 2 V | 0 ≤ \|V\| ≤ 2.2 V (2 V for VMU) | 2 µV | 20 µV | 200 µV | 2 mV |
| 20 V | 0 ≤ \|V\| ≤ 22 V (20 V for VMU) | 20 µV | 200 µV | 2 mV | 20 mV |
| 40 V | 0 ≤ \|V\| ≤ 44 V | 40 µV | 400 µV | 4 mV | 40 mV |
| 100 V | 0 ≤ \|V\| ≤ 100 V (110 V for HPSMU) | 100 µV | 1 mV | 10 mV | 100 mV |
| 200 V (for HPSMU) | 0 ≤ \|V\| ≤ 200 V | 200 µV | 2 mV | 20 mV | 200 mV |

a. This column is applied to the sampling measurement that *initial interval* is set to 480 µs or shorter.

**Table 2-6 SMU output current, resolution, and compliance by range**

| Output range (actually used) | Output current [a] | Setting [b] resolution | Maximum *comp* value [c] | | |
|---|---|---|---|---|---|
| | | | HRSMU | MPSMU | HPSMU |
| 10 pA (for HRSMU) | 0 ≤ \|I\| ≤ 10 pA | 10 fA | ±100 V | - | - |
| 100 pA (for HRSMU) | 0 ≤ \|I\| ≤ 100 pA | 10 fA | ±100 V | - | - |
| 1 nA | 0 ≤ \|I\| ≤ 1 nA | 100 fA | ±100 V | ±100 V | ±200 V |
| 10 nA | 0 ≤ \|I\| ≤ 10 nA | 1 pA | ±100 V | ±100 V | ±200 V |
| 100 nA | 0 ≤ \|I\| ≤ 100 nA | 10 pA | ±100 V | ±100 V | ±200 V |
| 1 µA | 0 ≤ \|I\| ≤ 1 µA | 100 pA | ±100 V | ±100 V | ±200 V |
| 10 µA | 0 ≤ \|I\| ≤ 10 µA | 1 nA | ±100 V | ±100 V | ±200 V |
| 100 µA | 0 ≤ \|I\| ≤ 100 µA | 10 nA | ±100 V | ±100 V | ±200 V |
| 1 mA | 0 ≤ \|I\| ≤ 1 mA | 100 nA | ±100 V | ±100 V | ±200 V |
| 10 mA | 0 ≤ \|I\| ≤ 10 mA | 1 µA | ±100 V | ±100 V | ±200 V |
| 100 mA | 0 ≤ \|I\| ≤ 20 mA | 10 µA | ±100 V | ±100 V | ±200 V |
| | 20 mA < \|I\| ≤ 50 mA | | ±40 V | ±40 V | ±200 V |
| | 50 mA < \|I\| ≤ 100 mA | | ±20 V | ±20 V | ±100 V |
| 1 A (for HPSMU) | 0 ≤ \|I\| ≤ 50 mA | 100 µA | - | - | ±200 V |
| | 50 mA < \|I\| ≤ 125 mA | | - | - | ±100 V |
| | 125 mA < \|I\| ≤ 500 mA | | - | - | ±40 V |
| | 500 mA < \|I\| ≤ 1 A | | - | - | ±20 V |

a. Parameter name may be *base*, *bias*, *peak*, *stress*, *value*, *start*, *stop*, and so on.
b. Minimum resolution is *range* × 5E-5. However, the setting accuracy is not guaranteed for the resolution less than the value shown in the table.
c. This column shows the maximum value of the voltage compliance.

**Table 2-7 SMU measurement current and resolution by range**

| Measurement range (actually used) | Measurement current | Measurement resolution | | | |
|---|---|---|---|---|---|
| | | Integration time | | | High speed [a] sampling measurement |
| | | 1PLC or longer | 640 μs to 1.92 ms | 80 μs to 560 μs | |
| 10 pA (for HRSMU) | 0 ≤ \|I\| ≤ 10.5 pA | 1 fA | 1 fA | 1 fA | 10 fA |
| 100 pA (for HRSMU) | 0 ≤ \|I\| ≤ 115 pA | 1 fA | 1 fA | 10 fA | 100 fA |
| 1 nA | 0 ≤ \|I\| ≤ 1.15 nA | 10 fA | 10 fA | 100 fA | 1 pA |
| 10 nA | 0 ≤ \|I\| ≤ 11.5 nA | 10 fA | 100 fA | 1 pA | 10 pA |
| 100 nA | 0 ≤ \|I\| ≤ 115 nA | 100 fA | 1 pA | 10 pA | 100 pA |
| 1 μA | 0 ≤ \|I\| ≤ 1.15 μA | 1 pA | 10 pA | 100 pA | 1 nA |
| 10 μA | 0 ≤ \|I\| ≤ 11.5 μA | 10 pA | 100 pA | 1 nA | 10 nA |
| 100 μA | 0 ≤ \|I\| ≤ 115 μA | 100 pA | 1 nA | 10 nA | 100 nA |
| 1 mA | 0 ≤ \|I\| ≤ 1.15 mA | 1 nA | 10 nA | 100 nA | 1 μA |
| 10 mA | 0 ≤ \|I\| ≤ 11.5 mA | 10 nA | 100 nA | 1 μA | 10 μA |
| 100 mA | 0 ≤ \|I\| ≤ 100 mA (115 mA for HPSMU) | 100 nA | 1 μA | 10 μA | 100 μA |
| 1 A (for HPSMU) | 0 ≤ \|I\| ≤ 1 A | 1 μA | 10 μA | 100 μA | 1 mA |

a. This column is applied to the sampling measurement that *initial interval* is set to 480 μs or shorter.

**Table 2-8** **PGU output voltage, resolution, and compliance by range**

| Output range (actually used) | Output voltage [a] | Setting resolution | *comp* value [b] |
|---|---|---|---|
| 20 V | $0 \leq \lvert V \rvert \leq 20$ V | 4 mV | ±100 mA |
| 40 V | $0 \leq \lvert V \rvert \leq 40$ V | 8 mV | ±100 mA |

a. Parameter name may be *base*, *bias*, *peak*, *stress*, *value*, *start*, *stop*, and so on.
b. Set any value (e.g. 0.1) though the compliance is always 100 mA.

**Table 2-9** **PGU pulse period, pulse width, and delay time**

| *period* value | *width* value | *delay* value [a] | Resolution |
|---|---|---|---|
| 2.0 µs to 100.0 µs | 1.0 µs to 99.9 µs | 0 to 100.0 µs | 0.1 µs |
| 100 µs to 1000 µs | 1 µs to 999 µs | 0 to 1000 µs | 1 µs |
| 1.00 ms to 10.00 ms | 0.01 ms to 9.99 ms | 0 to 10.00 ms | 10 µs |
| 10.0 ms to 100.0 ms | 0.1 ms to 99.9 ms | 0 to 100.0 ms | 100 µs |
| 100 ms to 1000 ms | 1 ms to 999 ms | 0 to 1000 ms | 1 ms |
| 1.00 s to 10.00 s | 0.01 s to 9.99 s | 0 to 10.00 s | 10 ms |

a. The value must be $0 \leq$ *delay* $\leq$ *period* value.

**Table 2-10** **PGU leading transition time and trailing transition time**

| Leading transition time (*rise*) | Trailing transition time (*fall*) | Resolution |
|---|---|---|
| 100 ns to 1000 ns | 100 ns to 1000 ns | 1 ns |
| 0.50 µs to 10.00 µs | 0.50 µs to 10.00 µs | 10 ns |
| 5.0 µs to 100.0 µs | 5.0 µs to 100.0 µs | 100 ns |
| 50 µs to 1000 µs | 50 µs to 1000 µs | 1 µs |
| 0.5 ms to 10.00 ms | 0.5 ms to 10.00 ms | 10 µs |

# Status Code

After measurement is performed, the Agilent 4155/4156 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 2-11.

- “hp4156b_spotMeas”
- “hp4156b_measureM”
- “hp4156b_measureP”
- “hp4156b_sweepIv”
- “hp4156b_sweepMiv”
- “hp4156b_sweepPiv”
- “hp4156b_sweepPbias”
- “hp4156b_sample”
- “hp4156b_stress”
- “hp4156b_readData”

**Table 2-11** **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. |
| 16 | 5 | PGU reached its compliance. |

**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).

## hp4156b_abortMeasure

This function aborts the 4155/4156's present operation, such as the measurement executed by the hp4156b_startMeasure function, the pulse output by the hp4156b_forcePulse function, the stress force by the hp4156b_stress function, and so on.

**Syntax**

ViStatus _VI_FUNC hp4156b_abortMeasure(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_addSampleSyncIv

This function specifies dc voltage/current source and sets the parameters. The dc source is used for the sampling measurements. Source output starts at the beginning of the sampling measurement (beginning of the hold time), and stops at the end of the last sampling measurement point.

Sampling measurement channels are defined by the hp4156b_sample function, and sampling measurement timing is defined by the hp4156b_setSample function.

**Syntax**

ViStatus _VI_FUNC hp4156b_addSampleSyncIv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 base, ViReal64 bias, ViReal64 comp);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the source unit. 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 27 (PGU1), or 28 (PGU2). |
| mode | Source output mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Output ranging mode. 0 (auto) or positive value (limited auto). |
| base | Source output value before measurement trigger (in A or V). |
| bias | Source output value after measurement trigger (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. |

**NOTE** range, base, bias, comp parameters

Available values depend on the unit. See "Parameters" on page 2-6.

## hp4156b_addSampleSyncPulse

This function specifies pulse voltage source and sets the parameters. The pulse source is used for the sampling measurements. Pulse outputs start at the beginning of the sampling measurement (beginning of the hold time), and stop at the end of the last sampling measurement point or stop at the last pulse if it comes earlier than the last sampling measurement point.

Sampling measurement channels are defined by the hp4156b_sample function, and sampling measurement timing is defined by the hp4156b_setSample function.

If you want to let the pulse output synchronize with the sampling measurement timing, you should define carefully both the hp4156b_addSampleSyncPulse timing parameters (count, period, width, delay, rise and fall) and the hp4156b_setSample timing parameters.

**Syntax**

ViStatus _VI_FUNC hp4156b_addSampleSyncPulse(ViSession vi, ViInt32 channel, ViReal64 base, ViReal64 peak, ViInt32 count, ViReal64 period, ViReal64 width, ViReal64 delay, ViReal64 rise, ViReal64 fall);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the pulse source. 27 (PGU1) or 28 (PGU2). |
| base | Pulse base value (in V). See Table 2-8. |
| peak | Pulse peak value (in V). See Table 2-8. |
| count | Pulse count (number of pulses, 1 to 65535) or 0 (pulse output free run mode). |
| period | Pulse period (in seconds). See Table 2-9. |
| width | Pulse width (in seconds). See Table 2-9. |
| delay | Pulse delay time (in seconds). See Table 2-9. |
| rise | Pulse leading transition time (in seconds). See Table 2-10. |
| fall | Pulse trailing transition time (in seconds). See Table 2-10. |

## hp4156b_addStressSyncIv

This function specifies dc stress source and sets the parameters. Up to 4 stress sources can be used at once. Use this function and/or hp4156b_addStressSyncPulse to set the stress sources.

**Syntax**

ViStatus _VI_FUNC hp4156b_addStressSyncIv(ViSession vi, ViInt32 source, ViInt32 channel, ViInt32 mode, ViReal64 range, ViReal64 base, ViReal64 stress, ViReal64 comp);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| source | Reference number of the stress source. 1, 2, 3, or 4. |
| channel | Channel number of the stress source. 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 27 (PGU1), or 28 (PGU2). |
| mode | Source output mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Output ranging mode. 0 (auto) or positive value (limited auto). |
| base | Source output value before stress start trigger (in A or V). |
| stress | Source output value after stress start trigger (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. |

**NOTE**

range, base, stress, comp parameters

Available values depend on the unit. See “Parameters” on page 2-6.

## hp4156b_addStressSyncPulse

This function specifies pulse stress source and sets the parameters. Up to 4 stress sources can be used at once. Use this function and/or hp4156b_addStressSyncIv to set the stress sources.

**Syntax**

ViStatus _VI_FUNC hp4156b_addStressSyncPulse(ViSession vi, ViInt32 source, ViInt32 channel, ViReal64 base, ViReal64 stress, ViReal64 width, ViReal64 delay, ViReal64 rise, ViReal64 fall);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| source | Reference number of the stress source. 1, 2, 3, or 4. |
| channel | Channel number of the pulse source. 27 (PGU1) or 28 (PGU2) |
| base | Stress pulse base value (in V). See Table 2-8. |
| stress | Stress pulse peak value (in V). See Table 2-8. |
| width | Pulse width (in seconds). See Table 2-9. |

| | |
|---|---|
| delay | Pulse delay time (in seconds). See Table 2-9. |
| rise | Pulse leading transition time (in seconds). See Table 2-10. |
| fall | Pulse trailing transition time (in seconds). See Table 2-10. |

## hp4156b_autoCal

This function enables or disables the auto calibration function.

**Syntax** ViStatus _VI_FUNC hp4156b_autoCal(ViSession vi, ViInt32 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| state | Auto calibration mode. 0 (off) or 1 (on). |

## hp4156b_clearSampleSync

This function clears the channel setup defined by the hp4156b_addSampleSyncIv function and the hp4156b_addSampleSyncPulse function.

**Syntax** ViStatus _VI_FUNC hp4156b_clearSampleSync(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_clearStressSync

This function clears the channel setup defined by the hp4156b_addStressSyncIv function and the hp4156b_addStressSyncPulse function.

**Syntax** ViStatus _VI_FUNC hp4156b_clearStressSync(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_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 hp4156b_close(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_cmd

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

**Syntax**

ViStatus _VI_FUNC hp4156b_cmd(ViSession vi, ViString cmd_str);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |

**Example**

```
ViSession vi;
ViStatus ret;
ret = hp4156b_cmd(vi, "AB");   /* sends the AB command */
```

## hp4156b_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 hp4156b_cmdData_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Length of result in bytes. 2 to 32767. |
| result[ ] | Response from instrument. |

## hp4156b_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 hp4156b_cmdInt(ViSession vi, ViString cmd_str, ViInt32 value);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |

| | |
|---|---|
| value | Parameter for command. -2147483647 to 2147483647. |

## hp4156b_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 hp4156b_cmdInt16Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt16 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. Returned data. |

## hp4156b_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 hp4156b_cmdInt16_Q(ViSession vi, ViString cmd_str, ViPInt16 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hp4156b_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 hp4156b_cmdInt32Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt32 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. Returned data. |

## hp4156b_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 hp4156b_cmdInt32_Q(ViSession vi, ViString cmd_str, ViPInt32 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hp4156b_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 hp4156b_cmdReal(ViSession vi, ViString cmd_str, ViReal64 value);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| value | Parameter for command. -1E+300 to 1E+300. |

## hp4156b_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 hp4156b_cmdReal64Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViReal64 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. Returned data. |

## hp4156b_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 hp4156b_cmdReal64_Q(ViSession vi, ViString cmd_str, ViPReal64 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hp4156b_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 hp4156b_cmdString_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Length of result in bytes. 2 to 32767. |
| result[ ] | Response from instrument. |

## hp4156b_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 hp4156b_dcl(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_error_message

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

**Syntax**

ViStatus _VI_FUNC hp4156b_error_message(ViSession vi, ViStatus error_number, ViChar _VI_FAR message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| error_number | Error code from the driver function. |
| message[ ] | Error message string. Returned data. This is limited to 256 characters. |

## hp4156b_error_query

This function returns the error codes and corresponding error messages in the error queue of an instrument. See *If You Have a Problem* manual for a listing of the instrument error codes 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 hp4156b_errorQueryDetect function.

**Syntax**

ViStatus _VI_FUNC hp4156b_error_query(ViSession vi, ViPInt32 error_number, ViChar _VI_FAR error_message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

| | |
|---|---|
| error_number | Instrument's error code. Returned data. |
| error_message[ ] | Instrument's error message. Returned data. Up to 256 characters. |

## hp4156b_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 hp4156b_errorQueryDetect(ViSession vi, ViBoolean errorQueryDetect);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| errorQueryDetect | Error checking enable (VI_TRUE) or disable (VI_FALSE). |

## hp4156b_errorQueryDetect_Q

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

**Syntax**

ViStatus _VI_FUNC hp4156b_errorQueryDetect_Q(ViSession vi, ViPBoolean pErrDetect);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| pErrDetect | Error checking enable (VI_TRUE) or disable (VI_FALSE). Returned data. |

## hp4156b_esr_Q

This function returns the contents of the ESR register. The driver returns the equivalent messages.

**Syntax**

ViStatus _VI_FUNC hp4156b_esr_Q(ViSession vi, ViChar _VI_FAR errstr[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| errstr[ ] | Response from instrument. 1 (ESR_OPC), 2 (ESR_RQL), 4 (ESR_QYE), 8 (ESR_DDE), 16 (ESR_EXE), 32 (ESR_CME), 64 (ESR_URQ), or 128 (ESR_PON). |

## hp4156b_execCal

This function executes the calibration and returns the calibration result. The parameter “result” returns the calibration result.

**Syntax**

ViStatus _VI_FUNC hp4156b_execCal(ViSession vi, ViPInt32 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| result | Calibration result. Returned data. Numeric number.<br>0: No error (calibration succeed). |

## hp4156b_execOffsetCancel

This function measures the zero offset data, and sets the zero offset function to on.

The parameter ‘channel’ specifies the measurement channel (SMU or VMU).
If you define SMU for ‘channel’, the SMU must be set to the voltage force mode by using the hp4156b_force function, before executing this function.

If you define VMU for ‘channel’, the VMU must be set to the differential voltage measurement mode by using the hp4156b_setVm function, before executing this function.

**Syntax**

ViStatus _VI_FUNC hp4156b_execOffsetCancel(ViSession vi, ViInt32 channel, ViInt32 range);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the unit to measure the zero offset data. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| range | Measurement range to measure the zero offset data. 0, 1, 2, or 3.<br>0: 10 pA range for SMU<br>1: 100 pA range for SMU<br>2: 1 nA range for SMU<br>3: 0.2 V range for VMU |

## hp4156b_force

This function specifies dc current/voltage source and forces the specified output immediately. To stop the output, use the hp4156b_force function with zero output.

**Syntax**

ViStatus _VI_FUNC hp4156b_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 “Parameters” on page 2-6.

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the source unit. 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 27 (PGU1), or 28 (PGU2). |
| mode | Source output mode. 1 (current, only for SMU) 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 (only for SMU). 0 (auto) or 1 (manual).<br>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.<br>If *polarity*=1, the specified *comp* polarity is kept. |

## hp4156b_forcePulse

This function specifies pulse voltage source and forces the specified pulse immediately. To stop the pulse output, use hp4156b_abortMeasure function.

**Syntax**

ViStatus _VI_FUNC hp4156b_forcePulse(ViSession vi, ViInt32 channel, ViInt32 count, ViReal64 base, ViReal64 peak, ViReal64 width, ViReal64 period, ViReal64 delay, ViReal64 rise, ViReal64 fall);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the pulse source. 27 (PGU1) or 28 (PGU2). |
| count | Pulse count (number of pulses, 1 to 65535) or 0 (pulse output free run mode). |
| base | Pulse base value (in V). See Table 2-8. |
| peak | Pulse peak value (in V). See Table 2-8. |

| | |
|---|---|
| width | Pulse width (in seconds). See Table 2-9. |
| period | Pulse period (in seconds). See Table 2-9. |
| delay | Pulse delay time (in seconds). See Table 2-9. |
| rise | Pulse leading transition time (in seconds). See Table 2-10. |
| fall | Pulse trailing transition time (in seconds). See Table 2-10. |

**NOTE**

**Pulse Count and Pulse Period**

Pulse count (*count*) and pulse period (*period*) settings are effective for both PGU1 and PGU2. If you use a PGU as a pulse source set by this function and another PGU as a stress source set by the hp4156b_setStress function, check the setting of these parameters. The parameter values must be the same as shown below.

hp4156b_forcePulse's *count* = hp4156b_setStress's *duration* (pulse count)

hp4156b_forcePulse's *period* = hp4156b_setStress's *period*

## hp4156b_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.

This function is not required for the Visual Basic .NET and Agilent VEE environments.

If the hp4156b_init function encounters an error, then the value of the vi output parameter will be VI_NULL.

### Syntax

ViStatus _VI_FUNC hp4156b_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. |

# hp4156b_measureM

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

**Syntax**

ViStatus _VI_FUNC hp4156b_measureM(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViReal64 value[ ], ViInt32 status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel[ ] | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2).<br><br>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 2-2 and Table 2-5 or Table 2-7. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

**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 = hp4156b_setSwitch(vi, mch[0], 1);
ret = hp4156b_setSwitch(vi, mch[1], 1);
ret = hp4156b_force(vi, mch[0], vmode, 0, 0, 0.1, 0);
ret = hp4156b_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];     /* measurement value */
ViInt32 st[2];      /* measurement status */
ret = hp4156b_measureM(vi, mch, mode, range, &md[0], &st[0]);
```

## hp4156b_measureP

This function executes a pulsed spot measurement by the specified channel, and returns the measured value and the measurement status.

**Syntax**

ViStatus _VI_FUNC hp4156b_measureP(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPReal64 value, ViPInt32 status);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| mode | Measurement mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| value | Measurement data. Returned data. |
| status | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

## hp4156b_offsetCancel

This function enables or disables the zero offset cancel function.

**Syntax**

ViStatus _VI_FUNC hp4156b_offsetCancel(ViSession vi, ViInt32 channel, ViInt32 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the unit to set the offset cancel function. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| state | Function state. 0 (off) or 1 (on). |

## hp4156b_opc_Q

This function does the *OPC? common command.

**Syntax**

ViStatus _VI_FUNC hp4156b_opc_Q(ViSession vi, ViPBoolean result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_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 = hp4156b_opc_Q(vi,&result);
```

## hp4156b_readData

This function reads and returns the source setup data or the data measured by the hp4156b_startMeasure function.

**Syntax**

ViStatus _VI_FUNC hp4156b_readData(ViSession vi, ViPInt32 eod, ViPInt32 data_type, ViPReal64 value, ViPInt32 status, ViPInt32 channel);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| eod | End of data flag. Returned data. 1 (end of data) or 0 (not EOD). |
| data_type | Data type of “value”. Returned data. 0, 1, 3, 8, 9, 11, 14, or 15.<br>0: Voltage setup data<br>1: Current setup data<br>3: Time setup data<br>8: Voltage measurement data<br>9: Current measurement data<br>11: Time measurement data<br>14: Sampling index data<br>15: Stress status data |
| value | Measurement data or source setup data. Returned data. |
| status | Measurement status or source status. Returned data. For the status value, see “Status Code” on page 2-12. |
| channel | Channel number of the unit for measurement or output. Returned data. 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 23 (VMU1), 24 (VMU2), 27 (PGU1), or 28 (PGU2). |

## hp4156b_readStatusByte_Q

This function returns the contents of the status byte register.

**Syntax** ViStatus _VI_FUNC hp4156b_readStatusByte_Q(ViSession vi, ViPInt16 statusByte);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| statusByte | Contents of the status byte. Returned data. |

## hp4156b_recoverOutput

This function returns the specified channel to the settings that are stored by the hp4156b_zeroOutput function, and clears the stored settings.

**Syntax** ViStatus _VI_FUNC hp4156b_recoverOutput(ViSession vi, ViInt32 channel);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the unit to return the settings. 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 27 (PGU1), or 28 (PGU2). |

## hp4156b_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 hp4156b_dcl function.

**Syntax** ViStatus _VI_FUNC hp4156b_reset(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

## hp4156b_revision_query

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

**Syntax** ViStatus _VI_FUNC hp4156b_revision_query(ViSession vi, ViChar_VI_FAR driver_rev[ ], ViChar _VI_FAR instr_rev[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| driver_rev[ ] | Instrument driver revision. Returned data. This is limited to 256 characters. |
| instr_rev[ ] | Instrument firmware revision. Returned data. This is limited to 256 characters. |

## hp4156b_sample

This function executes a sampling measurement by the specified channels, and returns the number of measurement points, measurement data index, measurement data, and the measurement status.

Before executing this function, set the sampling timing by the hp4156b_setSample function. The synchronous dc sources used with the sampling measurement units are defined by the hp4156b_addSampleSyncIv function. And the synchronous pulsed sources used with the sampling measurement units are defined by the hp4156b_addSampleSyncPulse function.

**Syntax**

ViStatus _VI_FUNC hp4156b_sample(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViPInt32 point, ViInt32 index[ ], ViReal64 value[ ], ViInt32 status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel[ ] | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2).<br><br>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, only for SMU) or 2 (voltage). |
| range[ ] | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| point | Number of measurement points. Returned data. |
| index[ ] | Measurement data index. Returned data. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

## hp4156b_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 hp4156b_self_test(ViSession vi, ViPInt16 test_result, ViChar_VI_FAR test_message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_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. This is limited to 256 characters. |

## hp4156b_setFilter

This function sets the output filter of the specified SMU.

**Syntax**

ViStatus _VI_FUNC hp4156b_setFilter(ViSession vi, ViInt32 channel, ViInt32 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the unit. 1 (SMU1), 2, 3, 4, 5, or 6 (SMU6). |
| state | SMU filter state. 0 (off) or 1 (on). |

## hp4156b_setInteg

This function sets the integration time and sets the number of samples that are taken and averaged for the measurement.

**Syntax**

ViStatus _VI_FUNC hp4156b_setInteg(ViSession vi, ViInt32 table, ViReal64 time, ViInt32 average);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| table | Integration time table. 1 (short), 2 (medium), or 3 (long). |
| time | Integration time (in seconds).<br>For *table*=1: 80 µs to 10.16 ms.<br>For *table*=2: Any value is ok (e.g. 1). The value will be ignored.<br>For *table*=3: 16.7 ms to 2.0 s |
| average | Number of samples for averaging, 1 to 1023. Or enter 0 to use the last setting. |

# hp4156b_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 hp4156b_setPbias function.

## Syntax

ViStatus _VI_FUNC hp4156b_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 “Parameters” on page 2-6.

## Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the sweep source. 1 to 6 (SMU1 to SMU6), 21 (VSU1), or 22 (VSU2). |
| mode | Output mode. 1, 2, 3, 4, -1, -2, -3, or -4. For the log sweep mode, *start* and *stop* must be the same polarity.<br>1: Voltage-Single-Linear sweep<br>2: Voltage-Single-Log sweep<br>3: Voltage-Double-Linear sweep<br>4: Voltage-Double-Log sweep<br>-1: Current-Single-Linear sweep (only for SMU)<br>-2: Current-Single-Log sweep (only for SMU)<br>-3: Current-Double-Linear sweep (only for SMU)<br>-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.<br>0.001 to 2.0 VA (for 4155/4156 and MPSMU in 41501)<br>0.001 to 14.0 VA (for HPSMU in 41501)<br>Setting resolution: 0.001 VA |

## hp4156b_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.

Filter of the pulse source must be set to off by using the hp4156b_setFilter function.

For the staircase sweep with pulsed bias measurements, the pulse output synchronizes with the staircase sweep output by the hp4156b_setIv function.

### Syntax

ViStatus _VI_FUNC hp4156b_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 “Parameters” on page 2-6.

### Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the pulse source. 1 to 6 (SMU1 to SMU6), 21 (VSU1), or 22 (VSU2). |
| mode | Pulse output mode. 1 (current, only for SMU) 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). 0.5 ms to 0.1 s. 0.1 ms resolution. |
| period | Pulse period (in seconds). 5 ms to 1.0 s. 0.1 ms resolution.<br>The value must be *width* + 4 ms or more. |
| hold | Hold time (in seconds). 0.0 to 655.35 s. 0.01 s 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.<br>For *mode*= 1, if *base* or *peak* is within 10 μA and not 0, *comp* must be 2 V or less.<br>For *mode*= 2, the minimum *comp* value must be as shown in Table 2-12. |

**Table 2-12 Minimum Compliance Value for Voltage Pulse Output**

| Voltage pulse [a] | *comp* (current compliance) |
|---|---|
| 0 < \|Vp-p\| < 2 V | \|*comp*\| > 2 nA |
| 2 V < \|Vp-p\| < 20 V | \|*comp*\| > \|Vp-p\|×1.111×10E−6 −2.22×10E−6 |
| 20 V < \|Vp-p\| | \|*comp*\| > 20 μA |

a. |Vp-p| is the voltage from the pulse base value to the pulse peak value.

## hp4156b_setPguR

This function sets the PGU output impedance.

**Syntax**

ViStatus _VI_FUNC hp4156b_setPguR(ViSession vi, ViInt32 channel, ViInt32 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of PGU. 27 (PGU1) or 28 (PGU2). |
| state | Status. 0 (approx. 0 ohm low impedance) or 1 (50 ohm). |

# hp4156b_setPiv

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

Filter of the pulse source must be set to off by using the hp4156b_setFilter function.

## Syntax

ViStatus _VI_FUNC hp4156b_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 “Parameters” on page 2-6.

## Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the pulse sweep source. 1 to 6 (SMU1 to SMU6), 21 (VSU1), or 22 (VSU2). |
| mode | Output mode. 1, 2, 3, 4, -1, -2, -3, or -4. For the current mode or log sweep mode, *base*, *start*, and *stop* must be the same polarity.<br>1: Voltage-Single-Linear sweep<br>2: Voltage-Single-Log sweep<br>3: Voltage-Double-Linear sweep<br>4: Voltage-Double-Log sweep<br>-1: Current-Single-Linear sweep (only for SMU)<br>-2: Current-Single-Log sweep (only for SMU)<br>-3: Current-Double-Linear sweep (only for SMU)<br>-4: Current-Double-Log sweep (only for SMU) |
| 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 s. 0.01 s resolution. |
| width | Pulse width (in seconds). 0.5 ms to 0.1 s. 0.1 ms resolution. |
| period | Pulse period (in seconds). 5 ms to 1.0 s. 0.1 ms resolution. The value must be *width* + 4 ms or more. |

| | |
|---|---|
| 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.<br><br>For the current output, if *base*, *start*, or *stop* is within 10 μA and not 0, *comp* must be 2 V or less.<br><br>For the voltage output, the minimum *comp* value must be as shown in Table 2-12. |

## hp4156b_setSample

This function specifies the measurement timing of the sampling measurements. The sampling measurement units are defined by the hp4156b_sample function.

**Syntax**

ViStatus _VI_FUNC hp4156b_setSample(ViSession vi, ViReal64 hold, ViReal64 interval, ViInt32 point);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| hold | Hold time (in seconds). -30 ms to 655.35 s. 100 μs resolution. |
| interval | Sampling interval (in seconds). 60 μs to 480 μs (20 μs resolution), 480 μs to 1 s (80 μs resolution), or 1 s to 65.534 s (2 ms resolution). |
| point | Number of sampling points. 1 to 10001. |

## hp4156b_setStress

This function sets the timing parameters of the stress.

**Syntax**

ViStatus _VI_FUNC hp4156b_setStress(ViSession vi, ViReal64 hold, ViInt32 mode, ViReal64 duration, ViReal64 period);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| hold | Hold time (in seconds). 0 to 655.35 s. |
| mode | Stress mode. 1 (pulse count mode) or 2 (duration mode). |
| duration | Meaning of this parameter depends on the mode setting:<br><br>If *mode*=1, this is the number of pulse count. 1 to 65535. |

If *mode*=2, this is the stress time (in seconds). 500 µs to 655.0 s.

period — Pulse period (in seconds). 2 µs to 10.0 s. Only for pulse stress.

The *period* value must match the *width* and *delay* values of the hp4156b_addStressSyncPulse function. See Table 2-9.

For *mode*=1, if you set the automatic abort function by using the hp4156b_stopMode function, the pulse output must be more than 10 seconds (*duration* × *period* > 10 s).

**NOTE**

**Pulse Count and Pulse Period**

Pulse count (*duration*) and pulse period (*period*) settings are effective for both PGU1 and PGU2. If you use a PGU as a stress source set by this function and another PGU as a pulse source set by the hp4156b_forcePulse function, check the setting of these parameters. The parameter values must be the same as shown below.

hp4156b_setStress's *duration* (pulse count) = hp4156b_forcePulse's *count*

hp4156b_setStress's *period* = hp4156b_forcePulse's *period*

## hp4156b_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 hp4156b_setIv or hp4156b_setPiv function must be executed before this function.

For the staircase sweep measurements, the output synchronizes with the staircase sweep output by the hp4156b_setIv function.

For the pulsed sweep measurements, the output synchronizes with the pulsed sweep output by the hp4156b_setPiv function.

For the staircase sweep with pulsed bias measurements, the output synchronizes the staircase sweep output by the hp4156b_setIv function and the pulse output by the hp4156b_setPbias function.

**Syntax**

```
ViStatus _VI_FUNC hp4156b_setSweepSync(ViSession vi, ViInt32 channel,
ViInt32 mode, ViReal64 range, ViReal64 start, ViReal64 stop, ViReal64 comp,
ViReal64 p_comp);
```

**Parameters**

vi — Instrument handle returned from hp4156b_init( ).

| | |
|---|---|
| channel | Channel number of the sweep source. 1 to 6 (SMU1 to SMU6), 21 (VSU1), or 22 (VSU2). |
| mode | Source output mode. 1 (current, only for SMU) or 2 (voltage).<br>Set 1 if the hp4156b_setIv or hp4156b_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. Available values are listed below. If you enter the other value, the power compliance is not set.<br>0.001 to 2.0 VA (for 4155/4156 and MPSMU in 41501)<br>0.001 to 14.0 VA (for HPSMU in 41501)<br>Setting resolution: 0.001 VA |

**NOTE**

range, start, stop, comp parameters

Available values depend on the unit. See “Parameters” on page 2-6.

Sweep type, linear or log, is set by the hp4156b_setIv or hp4156b_setPiv function. If the function sets the log sweep, *start* and *stop* must be the same polarity.

## hp4156b_setSwitch

This function sets the output switch of the specified channel.

**Syntax**

ViStatus _VI_FUNC hp4156b_setSwitch(ViSession vi, ViInt32 channel, ViInt32 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |

| | |
|---|---|
| channel | Channel number of the unit. 0 (all channels), 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 23 (VMU1), 24 (VMU2), 26 (GNDU), 27 (PGU1) or 28 (PGU2). |
| state | Output switch setting. 0 (off) or 1 (on). |

## hp4156b_setVm

This function sets VMU1 and VMU2 to the normal mode that can use two VMUs individually or the differential voltage measurement mode that uses two VMUs to measure voltage between two terminals.

**Syntax**

ViStatus _VI_FUNC hp4156b_setVm(ViSession vi, ViInt32 mode);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| mode | Measurement mode. 1 (normal) or 2 (differential). |

## hp4156b_spotMeas

This function executes a high speed spot measurement by the specified channel, and returns the measured value and the measurement status.

**Syntax**

ViStatus _VI_FUNC hp4156b_spotMeas(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPReal64 value, ViPInt32 status);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| mode | Measurement mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| value | Measurement data. Returned data. |
| status | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

# hp4156b_startMeasure

This function starts the specified measurement by the specified channels. You can read the measured data by using the hp4156b_readData function. The measurement data is entered to the 4155/4156 output buffer in the measurement order. If you want to abort the measurement, use the hp4156b_abortMeasure function.

## Syntax

ViStatus _VI_FUNC hp4156b_startMeasure(ViSession vi, ViInt32 meas_type, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViInt32 source);

## Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| meas_type | Measurement type. 1 (multi spot), 2 (staircase sweep), 3 (pulse spot), 4 (pulse sweep), 5 (sweep with pulsed bias), 10 (sampling), or 11 (stress force). |
| channel[ ] | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2).<br><br>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, only for SMU) or 2 (voltage). |
| range[ ] | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| source | Source data output mode. 0 (measurement data output without source data) or 1 (measurement data output with source data). |

## 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 = hp4156b_setSwitch(vi, mch[0], 1);
ret = hp4156b_setSwitch(vi, mch[1], 1);
ret = hp4156b_setFilter(vi, mch[0], 0);

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= hp4156b_setPbias(vi, mch[0], om, or, v1, v2, tw, tp, th, ic);
```

```
ret= hp4156b_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 = hp4156b_startMeasure(vi, type, mch, mode, range, 0);

ViInt32 eod;        /* eod */
ViInt32 dtype;      /* data type */
ViReal64 md;        /* measurement value */
ViInt32 st;         /* measurement status */
ViInt32 ch;         /* channel */
ret = hp4156b_readData(vi, &eod, &dtype, &md, &st, &ch);
printf("I1 = %9.6f mA \n", md * 1000);
ret = hp4156b_readData(vi, &eod, &dtype, &md, &st, &ch);
printf("I2 = %9.6f mA \n", md * 1000);
```

## hp4156b_stopMode

This function specifies the stop condition which enables the automatic abort function for the sweep measurement, sampling measurement, or stress force. Also this function specifies the sweep source output of the measurement unit after the sweep measurement is aborted.

### Syntax

ViStatus _VI_FUNC hp4156b_stopMode(ViSession vi, ViInt32 occ_stop, ViInt32 tcc_stop, ViInt32 ovf_stop, ViInt32 osc_stop, ViInt32 last_mode);

### Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| occ_stop | Automatic abort function by compliance of another unit. |
| | 0 (disables this abort mode) or 1 (enables this abort mode). |
| tcc_stop | Automatic abort function by compliance of this unit. |
| | 0 (disables this abort mode) or 1 (enables this abort mode). |
| ovf_stop | Automatic abort function by overflow of AD converter. |
| | 0 (disables this abort mode) or 1 (enables this abort mode). |
| osc_stop | Automatic abort function by oscillation of unit(s). |
| | 0 (disables this abort mode) or 1 (enables this abort mode). |
| last_mode | Source output value after abort condition. |
| | 1 (returns to start value), or 2 (keeps the value when aborted). |

## hp4156b_stress

This function forces the stress defined by the hp4156b_setStress, hp4156b_addStressSyncIv, and/or hp4156b_addStressSyncPulse functions.

**Syntax**

ViStatus _VI_FUNC hp4156b_stress(ViSession vi, ViPInt32 status);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| status | Stress output status. Returned data. For the status value, see “Status Code” on page 2-12. |

## hp4156b_sweepIv

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

Before executing this function, execute the hp4156b_setIv function to set the sweep source. Also, execute the hp4156b_setSweepSync function to set the synchronous sweep source.

**Syntax**

ViStatus _VI_FUNC hp4156b_sweepIv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| mode | Measurement mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| point | Number of measurement steps. Returned data. |
| source[ ] | Sweep source setup data. Returned data. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

**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 */
ViReal64 ts = 0.001;   /* 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 */

ret = hp4156b_setSwitch(vi, sch, 1);
ret = hp4156b_setSwitch(vi, mch, 1);
ret = hp4156b_force(vi, mch, om, or, v1, icomp, 0);
ret = hp4156b_setIv(vi, sch, sm, or, v1, v2, pts, th, td, ts,
icomp, pcomp);
ret = hp4156b_sweepIv(vi, mch, mm, mr, &mpts, &sc[0], &md[0],
&st[0]);
```

For the above example, the array variables sc[], md[], and st[] 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.

where, n = 0 to 10 (integer).

## hp4156b_sweepMiv

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

Before executing this function, execute the hp4156b_setIv function to set the sweep source. Also, execute the hp4156b_setSweepSync function to set the synchronous sweep source.

**Syntax**

ViStatus _VI_FUNC hp4156b_sweepMiv(ViSession vi, ViInt32 channel[ ], ViInt32 mode[ ], ViReal64 range[ ], ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel[ ] | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2).<br><br>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, only for SMU) or 2 (voltage). |
| range[ ] | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| point | Number of measurement steps. Returned data. |
| source[ ] | Sweep source setup data. Returned data. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

**Example**

```
ViSession vi;
ViStatus ret;
ViInt32 mch[3];       /* measurement channels */
mch[0] = 1;
mch[1] = 2;
mch[2] = 0;
ret = hp4156b_setSwitch(vi, mch[0], 1);
ret = hp4156b_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 = hp4156b_force(vi, mch[0], om, or, v1, icomp, 0);
ret = hp4156b_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 */
ret = hp4156b_sweepMiv(vi, mch, mm, mr, &mpts, &sc[0], &md[0],
&st[0]);
```

For the above example, the array variables sc[], md[], and st[] 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.

where, n = 0 to 10 (integer).

## hp4156b_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, and the measurement status.

Before executing this function, execute the hp4156b_setIv function and the hp4156b_setPbias function to set the staircase sweep source and the pulsed bias source respectively. Also, execute the hp4156b_setSweepSync function to set the synchronous sweep source.

### Syntax

ViStatus _VI_FUNC hp4156b_sweepPbias(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ]);

### Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| mode | Measurement mode. 1 (current, only for SMU) or 2 (voltage). |

| | |
|---|---|
| range | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| point | Number of measurement steps. Returned data. |
| source[ ] | Sweep source setup data. Returned data. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

**Example**

```
ViSession vi;
ViStatus ret;
ViInt32 pch = 1;       /* SMU1 for pulse source channel */
ret = hp4156b_setSwitch(vi, pch, 1);
ret = hp4156b_setFilter(vi, pch, 0);

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 = hp4156b_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 = hp4156b_setSwitch(vi, sch, 1);
ret = hp4156b_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 */
ret = hp4156b_sweepPbias(vi, sch, mm, mr, &mpts, &sc[0], &md[0],
&st[0]);
```

For the above example, the array variables sc[], md[], and st[] 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.

where, n = 0 to 10 (integer).

## hp4156b_sweepPiv

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

Before executing this function, execute the hp4156b_setPiv function to set the pulsed sweep source. Also, execute the hp4156b_setSweepSync function to set the synchronous sweep source.

### Syntax

ViStatus_VI_FUNC hp4156b_sweepPiv(ViSession vi, ViInt32 channel, ViInt32 mode, ViReal64 range, ViPInt32 point, ViReal64 source[ ], ViReal64 value[ ], ViInt32 status[ ]);

### Parameters

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the measurement unit. 1 to 6 (SMU1 to SMU6), 23 (VMU1), or 24 (VMU2). |
| mode | Measurement mode. 1 (current, only for SMU) or 2 (voltage). |
| range | Measurement ranging mode. 0 (auto), positive value (limited auto), or negative value (fixed range). For the available values, see Table 2-2 and Table 2-5 or Table 2-7. |
| point | Number of measurement steps. Returned data. |
| source[ ] | Sweep source setup data. Returned data. |
| value[ ] | Measurement data. Returned data. |
| status[ ] | Measurement status. Returned data. For the status value, see “Status Code” on page 2-12. |

### Example

```
ViSession vi;
ViStatus ret;
ViInt32 pch = 1;     /* SMU1 for pulse sweep source */
ret = hp4156b_setSwitch(vi, pch, 1);
ret = hp4156b_setFilter(vi, pch, 0);

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 = hp4156b_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 */
ret = hp4156b_sweepPiv(vi, pch, mm, mr, &mpts, &sc[0], &md[0],
&st[0]);
```

For the above example, the array variables sc[], md[], and st[] 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.

where, n = 0 to 10 (integer).

## hp4156b_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 hp4156b_timeOut(ViSession vi, ViInt32 timeOut);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| timeOut | I/O timeout value for all functions in the driver. in milliseconds. 0 to 2147483647. |

## hp4156b_timeOut_Q

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

**Syntax**

ViStatus_VI_FUNC hp4156b_timeOut_Q(ViSession vi, ViPInt32 pTimeOut);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| pTimeOut | Minimum timeout period that the driver can be set to, in milliseconds. |

## hp4156b_zeroOutput

This function stores the measurement setup of the units, and sets the units to 0 V output. To recover the setup, execute hp4156b_recoverOutput function.

**Syntax**

ViStatus_VI_FUNC hp4156b_zetoOutput(ViSession vi, ViInt32 channel);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hp4156b_init( ). |
| channel | Channel number of the unit to set to the zero output.<br>0 (all channels), 1 to 6 (SMU1 to SMU6), 21 (VSU1), 22 (VSU2), 27 (PGU1), or 28 (PGU2). |

# E5250A Driver Functions

This section provides the reference information of the Agilent E5250A VXI*plug&play* driver functions.

## Function List

Table 2-13 lists all the functions for the E5250A. You will see a brief description of the functions in the table.

For the description, syntax and parameters of the function, refer to the reference section following this table. The driver functions in the reference section will appear in the alphabetical order.

**Table 2-13 E5250A Driver Function List**

| Category | Function | Description |
|---|---|---|
| Miscellaneous | hpe5250a_init | Initializes the E5250A. |
| | hpe5250a_close | Closes the connection with the E5250A. |
| | hpe5250a_reset | Executes the E5250A reset. |
| | hpe5250a_self_test | Executes the E5250A self-test. |
| | hpe5250a_error_query | Queries for the E5250A error code/message. |
| | hpe5250a_error_message | Queries for the driver error. |
| | hpe5250a_revision_query | Queries for the E5250A firmware/driver revisions. |
| | hpe5250a_timeOut | Sets the timeout. |
| | hpe5250a_timeOut_Q | Queries for the timeout setting. |
| | hpe5250a_errorQueryDetect | Sets the automatic error checking. |
| | hpe5250a_errorQueryDetect_Q | Queries for the automatic error checking setting. |
| | hpe5250a_dcl | Sends the Device Clear. |
| | hpe5250a_esr_Q | Queries for the ESR status. |
| | hpe5250a_readStatusByte_Q | Reads the E5250A status byte. |
| | hpe5250a_opc_Q | Checks the E5250A operation completion status. |
| Mode Control | hpe5250a_func | Sets the configuration mode. |
| | hpe5250a_connRuleSeq | Sets the connection rule/sequence. |

| Category | Function | Description |
|---|---|---|
| Bias Mode | hpe5250a_biasPort | Selects the input bias port. |
| | hpe5250_biasChanCard | Selects the card for bias mode. |
| | hpe5250_biasChanList | Selects the channel list for bias mode. |
| | hpe5250a_biasState | Sets the bias port state. |
| | hpe5250a_biasChanList_Q | Queries for the bias channel list. |
| Couple Port | hpe5250a_couplePort | Selects the couple port. |
| | hpe5250a_coupleState | Sets the couple port state. |
| Route Control | hpe5250a_closeList | Closes the channel list. |
| | hpe5250a_openList | Opens the channel list. |
| | hpe5250a_openCard | Opens all output on the card. |
| | hpe5250a_closeList_Q | Queries for the channel list status. |
| | hpe5250a_openList_Q | |
| | hpe5250a_closeCard_Q | Queries for the closed channel list on the card. |
| C/G Compensation | hpe5250a_CompenC | Executes the C/G compensation. |
| | hpe5250a_selectCompenFile | Selects the compensation data file. |
| Diagnostics | hpe5250a_testExec_Q | Executes the relay/front-panel/controller test. |
| | hpe5250a_testClear | Clears the test result. |
| Passthrough Functions | hpe5250a_cmd | Sends a command. |
| | hpe5250a_cmdInt | Sends a command with an integer parameter. |
| | hpe5250a_cmdReal | Sends a command with a real parameter. |
| | hpe5250a_cmdData_Q | Sends a command to read any data. |
| | hpe5250a_cmdString_Q | Sends a command to read string response. |
| | hpe5250a_cmdInt16_Q | Sends a command to read 16 bit integer response. |
| | hpe5250a_cmdInt16Arr_Q | Sends a command to read 16 bit integer array response. |
| | hpe5250a_cmdInt32_Q | Sends a command to read 32 bit integer response. |
| | hpe5250a_cmdInt32Arr_Q | Sends a command to read 32 bit integer array response. |
| | hpe5250a_cmdReal64_Q | Sends a command to read 64 bit real response. |
| | hpe5250a_cmdReal64Arr_Q | Sends a command to read 64 bit real array response. |

## hpe5250a_biasChanCard

This function will enable or disable bias on all the output ports of the specified card.

**Syntax**

ViStatus _VI_FUNC hpe5250a_biasChanCard(ViSession vi,
ViInt16 disable_enable, ViInt16 bias_cardno);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| disable_enable | Bias status. 0 (sets bias enabled card) or 1 (sets bias disabled card). |
| bias_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |

## hpe5250a_biasChanList

This function will enable or disable bias on all the output ports specified by the biaschan_list.

The parameter 'biaschan_list' is an array of integers with each integer representing one channel. The last number of the 'biaschan_list' should be "0"(numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

**Syntax**

ViStatus _VI_FUNC hpe5250a_biasChanList(ViSession vi,
ViInt16 biaschan_disen, ViInt32 _VI_FAR biaschan_list[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| biaschan_disen | Bias status. 0 (sets bias enabled port) or 1 (sets bias disabled port). |
| biaschan_list[ ] | Channel numbers. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |

## hpe5250a_biasChanList_Q

This function will query the instrument for the bias status for the channels given in the list.

The parameter 'biaschan_list' is an array of integers with each integer representing one channel. The last number of the 'biaschan_list' should be "0" (numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

The 'bias_status' parameter is an array of integers containing the return values of the query. The 'bias_status' array returned will correspond one to one with 'biaschan_list' parameter.

**Syntax**

ViStatus _VI_FUNC hpe5250a_biasChanList_Q(ViSession vi, ViInt16 bias_disen, ViInt32 _VI_FAR biaschan_list[ ], ViInt32 _VI_FAR bias_status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| bias_disen | Bias status for the query. 0 (confirms if the port is the bias enabled) or 1 (confirms if the port is the bias disabled). |
| biaschan_list[ ] | Channel numbers to know the bias status. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |
| bias_status[ ] | Bias status of the channels given in the biaschan_list. Returned value depends on the setting of bias_disen as shown below:<br>when *bias_disen*=0, 0 means bias disabled, 1 means enabled.<br>when *bias_disen*=1, 0 means bias enabled, 1 means disabled. |

## hpe5250a_biasPort

This function will select which input port is the bias port on the specified card. For each card, you can specify the same or different Bias Port. This function applies only to the E5252A card.

**Syntax**

ViStatus _VI_FUNC hpe5250a_biasPort(ViSession vi, ViInt16 biasport_cardno, ViInt16 bias_port);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| biasport_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |

| | |
|---|---|
| bias_port | Input port number to be set to the bias port. 1 to 10 (input port 1 to input port 10). |

## hpe5250a_biasState

This function controls the bias mode for the specified card. When Bias Mode is on, the input Bias Port is connected to all bias enabled output ports that are not connected to any other input ports. Bias disabled output ports are never connected to the input Bias Port when Bias Mode is on.

**Syntax**

ViStatus _VI_FUNC hpe5250a_biasState(ViSession vi, ViInt16 biasstate_cardno, ViInt16 state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| biasstate_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |
| state | Bias mode. 0 (off) or 1 (on). |

## hpe5250a_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 hpe5250a_close(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

## hpe5250a_closeCard_Q

This function will query the card for the channels closed of the specified card.

The parameter ’closechan_list’ contains the channel numbers returned by the instrument. This will be an array of integers terminated by ’zero’ to identify the end of the list. Array of enough length should be passed to the function.

**Syntax**

ViStatus _VI_FUNC hpe5250a_closeCard_Q(ViSession vi, ViInt16 close_card, ViInt32 _VI_FAR closechan_list[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| close_card | Card number. 1 (card 1), 2 (card 2), 3 (card 3), or 4 (card 4) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |
| closechan_list[ ] | Channels closed of the specified card. |

## hpe5250a_closeList

This function will connect the input ports to the output ports specified by the channel list.

The parameter 'closechan_list' is an array of integers with each integer representing one channel. The last number of the 'closechan_list' should be "0" (numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

**Syntax**

ViStatus _VI_FUNC hpe5250a_closeList(ViSession vi,
ViInt32_VI_FAR closechan_list[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| closechan_list[ ] | Channel numbers to connect. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |

## hpe5250a_closeList_Q

This function will query the instrument for the channels closed given in the 'closechan_list'.

The parameter 'closechan_list' is an array of integers with each integer representing one channel. The last number of the 'closechan_list' should be "0" (numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

The 'close_status' parameter is an array of integers containing the return values of the query. The 'close_status' array returned will correspond one to one with 'closechan_list' parameter.

**Syntax**

ViStatus _VI_FUNC hpe5250a_closeList_Q(ViSession vi, ViInt32_VI_FAR closechan_list[ ], ViInt32 _VI_FAR close_status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| closechan_list[ ] | Channel numbers to know the close status. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |
| close_status[ ] | Status of the channels given in the closechan_list. 0 (opened) or 1 (closed). |

## hpe5250a_cmd

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

**Syntax**

ViStatus _VI_FUNC hpe5250a_cmd(ViSession vi, ViString cmd_str);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |

## hpe5250a_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 hpe5250a_cmdData_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Length of result in bytes. 2 to 32767. |
| result[ ] | Response from instrument. |

## hpe5250a_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 hpe5250a_cmdInt(ViSession vi, ViString cmd_str, ViInt32 value);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| value | Parameter for command. -2147483647 to 2147483647. |

## hpe5250a_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 hpe5250a_cmdInt16Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt16 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. |

## hpe5250a_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 hpe5250a_cmdInt16_Q(ViSession vi, ViString cmd_str, ViPInt16 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

| | |
|---|---|
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hpe5250a_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 hpe5250a_cmdInt32Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViInt32 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. |

## hpe5250a_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 hpe5250a_cmdInt32_Q(ViSession vi, ViString cmd_str, ViPInt32 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hpe5250a_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 hpe5250a_cmdReal(ViSession vi, ViString cmd_str, ViReal64 value);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| value | Parameter for command. -1E+300 to 1E+300. |

## hpe5250a_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 hpe5250a_cmdReal64Arr_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViReal64 _VI_FAR result[ ], ViPInt32 count);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Size of result[ ] (number of items in the array).<br>1 to 2147483647. |
| result[ ] | Response from instrument. |
| count | Count of valid items in result[ ]. |

## hpe5250a_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 hpe5250a_cmdReal64_Q(ViSession vi, ViString cmd_str, ViPReal64 result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| result | Response from instrument. |

## hpe5250a_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 hpe5250a_cmdString_Q(ViSession vi, ViString cmd_str, ViInt32 size, ViChar _VI_FAR result[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cmd_str | Instrument command (cannot exceed 256 bytes in length). |
| size | Length of result in bytes. 2 to 32767. |
| result[ ] | Response from instrument. |

## hpe5250a_compenC

This function compensates capacitance/conductance data measured by using Agilent 4284A C meter, and returns compensation results. If you change the compensation data, create the compensation data file, and specify the data file using hpe5250a_selectCompenFile function before executing this function.

**Syntax**

ViStatus _VI_FUNC hpe5250a_compenC(ViSession vi,ViReal64 frequency, ViReal64 len_hptrx, ViReal64 len_usrtrx_h, ViReal64 len_usrtrx_l, ViReal64 len_usrcoax_h, ViReal64 len_usrcoax_l, ViReal64 raw_c, ViReal64 raw_g, ViPReal64 compen_c, ViPReal64 compen_g);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| frequency | Measurement frequency. 1E3 to 1E6 Hz. |
| len_hptrx | Agilent 16494A triaxial cable. 1.5 or 3.0 m. |
| len_usrtrx_h | Triaxial cable length (in m) between connector plate and DUT high terminal. If you do not use triaxial cable, enter 0 (zero). |
| len_usrtrx_l | Triaxial cable length (in m) between connector plate and DUT low terminal. If you do not use triaxial cable, enter 0 (zero). |
| len_usrcoax_h | Coaxial cable length (in m) between connector plate and DUT high terminal. If you do not use coaxial cable, enter 0 (zero). |
| len_usrcoax_l | Coaxial cable length (in m) between connector plate and DUT low terminal. If you do not use coaxial cable, enter 0 (zero). |

| | |
|---|---|
| raw_c | Capacitance value (in F) measured by the 4284A. |
| raw_g | Conductance value (in S) measured by the 4284A. |
| compen_c | Capacitance compensation result (in F). |
| compen_g | Conductance compensation result (in S). |

**Example**

```
Sub Main()
 Dim Ag5250 As Hpe5250a = New Hpe5250a("GPIB0::22::INSTR", True,
True)
 Ag5250.Reset()
 Ag5250.TimeOut(60000)

 Dim closelist() As Integer = {40101, 40202, 40303, 0}
 'path: in1-out1, in2-out2, in3-out3 of card4
 Ag5250.Func(Hpe5250a.ChannelConfigEnum.NormalConfiguration)
 Ag5250.ConnRuleSeq(0, 1, 1)
 Ag5250.CloseList(closelist)

 Dim freq As Double = 1000000    ' measurement frequency: 1 (MHz)
 Dim c_len As Double = 3         ' 16494A cable length:   3 (m)
 Dim tri_h As Double = 0         ' HI side triax cable length in m
 Dim tri_l As Double = 0         ' LO side triax cable length in m
 Dim co_h As Double = 0          ' HI side coax cable length in m
 Dim co_l As Double = 0          ' LO side coax cable length in m
 Dim cr As Double = 0.0000000001 ' C value by 4284A: 100 (pF)
 Dim gr As Double = 0.0005       ' G value by 4284A: 0.5 (mS)
 Dim cc As Double
 Dim gc As Double
 Ag5250.CompenC(freq, c_len, tri_h, tri_l, co_h, co_l, cr, gr, cc,
gc)

 Dim result As String = "C = " & cc * 1000000000000.0 & " (pF)"
 result = result & Chr(10) & "G = " & gc * 1000 & " (mS)" & Chr(10)
 Console.WriteLine(result)

 Ag5250.Close()
End Sub
```

## hpe5250a_connRuleSeq

The function sets connection rule and connection sequence for the specified card.

**Syntax**

ViStatus _VI_FUNC hpe5250a_connRuleSeq(ViSession vi,
ViInt16 cardno_ruleseq, ViInt16 rule, ViInt16 sequence);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| cardno_ruleseq | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |

| | |
|---|---|
| rule | Connection rule. 0 (free route) or 1 (single route). |
| sequence | Connection sequence. 0 (no sequence), 1 (break before make), or 2 (make before break) |

## hpe5250a_couplePort

This function sets the couple ports which are used for making kelvin connections on the specified card. The specified input port number will be coupled with the next input port and two output ports. For each card, you may setup the same or different couple ports. This command overwrites the previous couple port setting for the card. This command applies only to the E5252A card.

The couple port mode is controlled by the hpe5250a_coupleState function.

**Syntax**

ViStatus _VI_FUNC hpe5250a_couplePort(ViSession vi,
ViInt16 coupleport_cardno, ViInt16 port1, ViInt16 port3, ViInt16 port5,
ViInt16 port7, ViInt16 port9);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| coupleport_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |
| port1 | Couple port by the input ports 1 and 2. 0 (disable) or 1 (enable). |
| port3 | Couple port by the input ports 3 and 4. 0 (disable) or 1 (enable). |
| port5 | Couple port by the input ports 5 and 6. 0 (disable) or 1 (enable). |
| port7 | Couple port by the input ports 7 and 8. 0 (disable) or 1 (enable). |
| port9 | Couple port by the input ports 9 and 10. 0 (disable) or 1 (enable). |

## hpe5250a_coupleState

This function controls the couple port mode for the specified card. This function applies only to the E5252A card.

**Syntax**

ViStatus _VI_FUNC hpe5250a_coupleState(ViSession vi,
ViInt16 couplestate_cardno, ViInt16 couple_state);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

| | |
|---|---|
| couplestate_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |
| couple_state | Couple port mode. 0 (off) or 1 (on). |

## hpe5250a_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 hpe5250a_dcl(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

## hpe5250a_error_message

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

**Syntax**

ViStatus _VI_FUNC hpe5250a_error_message(ViSession vi, ViStatus error_number, ViChar _VI_FAR message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| error_number | Error return value from the driver function. |
| message[ ] | Error message string. This is limited to 256 characters. |

## hpe5250a_error_query

This function returns the error numbers and corresponding error messages in the error queue of a instrument. See Agilent E5250A *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 hpe5250a_errorQueryDetect function.

**Syntax**

ViStatus _VI_FUNC hpe5250a_error_query(ViSession vi, ViPInt32 error_number, ViChar _VI_FAR error_message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| error_number | Instrument's error code. |
| error_message[ ] | Instrument's error message. This is limited to 256 characters. |

## hpe5250a_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 hpe5250a_errorQueryDetect(ViSession vi, ViBoolean errorQueryDetect);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| errorQueryDetect | Error checking enable (VI_TRUE) or disable (VI_FALSE). |

## hpe5250a_errorQueryDetect_Q

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

**Syntax**

ViStatus _VI_FUNC hpe5250a_errorQueryDetect_Q(ViSession vi, ViPBoolean pErrDetect);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| pErrDetect | Error checking enable (VI_TRUE) or disable (VI_FALSE). |

## hpe5250a_esr_Q

This function returns the contents of the ESR register. The driver returns the equivalent messages.

**Syntax** ViStatus _VI_FUNC hpe5250a_esr_Q(ViSession vi, ViChar _VI_FAR errstr[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| errstr[ ] | Response from instrument. 1, 2, 4, 8, 16, 32, 64, 128 or others.<br>1 (ESR_OPC), 2 (ESR_RQL), 4 (ESR_QYE_ERROR),<br>8 (ESR_DEVICE_DEPENDENT_ERROR),<br>16 (ESR_EXECUTION_ERROR),<br>32 (ESR_COMMAND_ERROR),<br>64 (ESR_URQ), 128 (ESR_PON), or<br>others (ESR_MULTI_EVENT). |

## hpe5250a_func

This function is used to set the channel configuration to the auto configuration mode or the normal configuration mode.

**Syntax** ViStatus _VI_FUNC hpe5250a_func(ViSession vi, ViInt16 channel_config);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| channel_config | Configuration mode. 0 (auto) or 1 (normal). |

## hpe5250a_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 hpe5250a_init function encounters an error, then the value of the vi output parameter will be VI_NULL.

**Syntax** ViStatus _VI_FUNC hpe5250a_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 In-System Verification), or<br>VI_FALSE (do not perform In-System Verification). |
| do_reset | VI_TRUE (to perform reset operation), or<br>VI_FALSE (do not perform reset operation). |

| | |
|---|---|
| vi | Instrument handle. This is VI_NULL if an error occurred during the init. |

## hpe5250a_opc_Q

This function does the *OPC? common command.

**Syntax**

ViStatus _VI_FUNC hpe5250a_opc_Q(ViSession vi, ViPBoolean result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| result | VI_TRUE (Operation complete), or<br>VI_FALSE (Operation is pending). |

## hpe5250a_openCard

This function will disconnect all input ports from all output ports for the specified card. Then if bias mode is on, connects the input bias port to all bias enabled output ports.

**Syntax**

ViStatus _VI_FUNC hpe5250a_openCard(ViSession vi, ViInt16 open_cardno);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| open_cardno | Card number. 1 (card 1), 2 (card 2), 3 (card 3), 4 (card 4), or 5 (all card) in the normal configuration mode, or 0 (all card in the automatic configuration mode). For the configuration mode, see hpe5250a_func. |

## hpe5250a_openList

This function will disconnect the input ports from the output ports specified by the channel list.

The parameter ’openchan_list’ is an array of integers with each integer representing one channel. The last number of the ’openchan_list’ should be “0” (numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

**Syntax**

ViStatus _VI_FUNC hpe5250a_openList(ViSession vi,
ViInt32_VI_FAR openchan_list[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

| | |
|---|---|
| openchan_list[ ] | Channel numbers to disconnect. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |

## hpe5250a_openList_Q

This function will query the instrument for the channels open given in the 'openchan_list'.

The parameter 'openchan_list' is an array of integers with each integer representing one channel. The last number of the 'openchan_list' should be "0" (numeric zero) to identify the end of the list. The maximum number of channels that can be specified by the list is 100.

The 'open_status' parameter is an array of integers containing the return values of the query. The 'open_status' array returned will correspond one to one with 'openchan_list' parameter.

**Syntax**

ViStatus _VI_FUNC hpe5250a_openList_Q(ViSession vi, ViInt32_VI_FAR openchan_list[ ], ViInt32 _VI_FAR open_status[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| openchan_list[ ] | Channel numbers to know the open status. 5 digits integer. ABCDE. where A: card number, BC: input port number, DE: output port number. Top zero(s) can be ignored. For example, if A=0, BC=01, and DE=01, channel number should be 101 instead of 00101. |
| open_status[ ] | Status of the channels given in the openchan_list. 1 (opened) or 0 (closed). |

## hpe5250a_readStatusByte_Q

This function returns the contents of the status byte register.

**Syntax**

ViStatus _VI_FUNC hpe5250a_readStatusByte_Q(ViSession vi, ViPInt16 statusByte);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| statusByte | The contents of the status byte are returned in this parameter. |

## hpe5250a_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 hpe5250a_dcl function.

**Syntax**

ViStatus _VI_FUNC hpe5250a_reset(ViSession vi);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |

## hpe5250a_revision_query

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

**Syntax**

ViStatus _VI_FUNC hpe5250a_revision_query(ViSession vi,
ViChar_VI_FAR driver_rev[ ], ViChar _VI_FAR instr_rev[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| driver_rev[ ] | Instrument driver revision. This is limited to 256 characters. |
| instr_rev[ ] | Instrument firmware revision. This is limited to 256 characters. |

## hpe5250a_selectCompenFile

This function specifies capacitance/conductance compensation data file used to compensate C/G by using hpe5250a_compenC.

**Syntax**

ViStatus _VI_FUNC hpe5250a_selectCompenFile(ViSession vi,
ViString file_name);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| file_name | Compensation data file name. Use absolute path. If the value is NULL string, the default data is used. |

**Remarks**

If you change the compensation data, copy the default data shown below, and modify the data for your measurement cable. You will need to change the data for DATA05 and 06, and/or DATA07 and 08 corresponding to your cables. To measure and change the compensation data, refer to Agilent E5250A *User's Guide*. To get

the R, L, and C value, measure R, L, and C of the cable using the 4284A, and divide them by cable length (in m). Compensation data must be the value for 1 m length. Do not change the data format in the file.

```
# E5250A   C Compensation coefficient data table
#
#   CAUTION : Do not add or delete "REVISION" line and "DATAxx" line.
#             Change the value for R,L,C of DATA05,06,07 or 08.
#
REVISION      A.03.00
#             R [ohm]       L [H]         C [F]
DATA00        74.65E-3      140.00E-9     58.44E-12     # Frame Path 1
DATA01        75.41E-3      90.00E-9      67.13E-12     # Frame Path 2
DATA02        231.41E-3     450.00E-9     178.85E-12    # Card Path High
DATA03        177.56E-3     390.00E-9     135.45E-12    # Card Path Low
DATA04        100.70E-3     400.00E-9     80.00E-12     # Triax Cable      [/m]
DATA05        100.70E-3     400.00E-9     80.00E-12     # User Triax Cbl H [/m]
DATA06        100.70E-3     400.00E-9     80.00E-12     # User Triax Cbl L [/m]
DATA07        114.00E-3     544.00E-9     130.00E-12    # User Coax Cbl H  [/m]
DATA08        114.00E-3     544.00E-9     130.00E-12    # User Coax Cbl L  [/m]
DATA09        0.00E-3       0.00E-9       1.20E-12      # Stray Capacitance
# END of Data
```

## hpe5250a_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 hpe5250a_self_test(ViSession vi, ViPInt16 test_result, ViChar_VI_FAR test_message[ ]);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| test_result | Numeric result from self-test operation. 0: No error. |
| test_message[ ] | Self-test status message. This is limited to 256 characters. |

## hpe5250a_testClear

This function clears the test result for the specified relay card or the front panel or the controller.

**Syntax**

ViStatus _VI_FUNC hpe5250a_testClear(ViSession vi, ViInt16 framecard_clear);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| framecard_clear | Test result to be cleared. 0 (test result of all test), 1 (card 1 relay test result), 2 (card 2 relay test result), 3 (card 3 relay test result), 4 (card 4 relay test result), 5 (relay test result of all card), 6 (front panel test result), or 7 (controller test result). |

## hpe5250a_testExec_Q

This function executes the controller test, the front panel test, or the relay test for the specified card. You must attach the relay test adapter before executing the relay test. The Front Panel test requires the key to be pressed within 10 seconds else the test will fail.

**Syntax**

ViStatus _VI_FUNC hpe5250a_testExec_Q(ViSession vi, ViInt16 framecard_exec, ViPInt16 exec_result);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| framecard_exec | Test to be executed. 1 (card 1 relay test) to 4 (card 4 relay test), 5 (relay test for all card), 6 (front panel test), or 7 (controller test). |
| exec_result | Test result. 0: No error. |

## hpe5250a_timeOut

This function sets a minimum timeout value for driver I/O transactions in milliseconds. The default timeout period is 2 seconds.

**Syntax**

ViStatus_VI_FUNC hpe5250a_timeOut(ViSession vi, ViInt32 timeOut);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| timeOut | I/O timeout value for all functions in the driver. in milliseconds. 0 to 2147483647. |

## hpe5250a_timeOut_Q

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

**Syntax**

ViStatus_VI_FUNC hpe5250a_timeOut_Q(ViSession vi, ViPInt32 pTimeOut);

**Parameters**

| | |
|---|---|
| vi | Instrument handle returned from hpe5250a_init( ). |
| pTimeOut | Minimum timeout period that the driver can be set to, in milliseconds. |

# 3 Programming Examples for Visual Basic Users

This chapter describes how to create measurement programs using the Agilent 4155/4156 and the 4155/4156 VXI*plug&play* driver, and provides programming examples using Microsoft Visual Basic. This chapter contains the following sections:

- “Programming Basics”
- “High-Speed Spot Measurements”
- “Multi-Channel Spot Measurements”
- “Staircase Sweep Measurements”
- “Synchronous Sweep Measurements”
- “Multi-Channel Sweep Measurements”
- “Pulsed Spot Measurements”
- “Multi-Channel Pulsed Spot Measurements”
- “Pulsed Sweep Measurements”
- “Multi-Channel Pulsed Sweep Measurements”
- “Staircase Sweep with Pulsed Bias Measurements”
- “Sampling Measurements”
- “Stress Force”

**NOTE**

## About Program Code

Programming examples are provided as subprograms that can be run with the project template shown in Table 3-1. The subprograms include the code to perform measurement, to display the measurement data, or to store the data. To execute the program, insert the subprograms instead of the perform_meas subprogram in the template.

# Programming Basics

This section provides the basic information for programming using the Agilent 4155/4156 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 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 4155/4156) to computer via GPIB.

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

**Step 3.** Import the following file to the project.

- hp4156b.bas (e.g. \Program Files\VISA\winnt\include\hp4156b.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 example code is written in Visual Basic 6.0.

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

**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.

**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 = hp4156b_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 = hp4156b_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 = hp4156b_reset(vi)                      'resets 4155/4156               '19
ret = hp4156b_timeOut(vi, 60000)             'sets time out to 60 sec
ret = hp4156b_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 = hp4156b_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 4155/4156. The above example is for the Agilent 4155/4156 on the GPIB address 17. Confirm the GPIB address of your 4155/4156, and set the address correctly instead of “17”. |
| 8 to 17 | Checks the status returned by the hp4156b_init function. If an error status is returned, displays a message box to show the error message, and stops the program execution. |
| 19 to 23 | Resets the Agilent 4155/4156, 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 = hp4156b_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 (hp4156b_INSTR_ERROR_DETECTED = ret) Then
    retStatus = hp4156b_error_query(vi, inst_err, err_message)
    msg = "Instrument Error:  " & inst_err & Chr(10) & Chr(10) & err_message
    MsgBox msg, vbOKOnly, ""
  Else
    retStatus = hp4156b_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 4155/4156. |
| 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 4155/4156, 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 4155/4156 VXI*plug&play* driver functions:

- hp4156b_setSwitch to enable/disable the source/measurement channels
- hp4156b_force, hp4156b_setIv, etc. to set source outputs
- hp4156b_spotMeas, hp4156b_sweepIv, etc. to perform measurements
- hp4156b_zeroOutput to disable source outputs

**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 Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform 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) = 3          ’SMU3: source
pins(3) = 4          ’SMU4: substrate

ret = hp4156b_setSwitch(vi, pins(3), 1)                  ’9
ret = hp4156b_setSwitch(vi, pins(2), 1)
ret = hp4156b_setSwitch(vi, pins(1), 1)
ret = hp4156b_setSwitch(vi, pins(0), 1)
check_err vi, ret                                        ’13

spot_meas vi, ret, pins()                                ’15

ret = hp4156b_setSwitch(vi, hp4156b_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 = hp4156b_force(vi, pins(3), hp4156b_VF_MODE, 0, 0, 0.05, 0)          '14
ret = hp4156b_force(vi, pins(2), hp4156b_VF_MODE, 0, 0, 0.05, 0)
ret = hp4156b_force(vi, pins(1), hp4156b_VF_MODE, 2, vg, igcomp, 0)
ret = hp4156b_force(vi, pins(0), hp4156b_VF_MODE, 2, vd, idcomp, 0)       '17
check_err vi, ret

ret = hp4156b_spotMeas(vi, pins(0), hp4156b_IM_MODE, 0, meas, status)     '20
check_err vi, ret
ret = hp4156b_zeroOutput(vi, hp4156b_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.<br>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 Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform 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 = hp4156b_setSwitch(vi, pins(2), 1)                   '8
ret = hp4156b_setSwitch(vi, pins(1), 1)
ret = hp4156b_setSwitch(vi, pins(0), 1)
check_err vi, ret                                         '11

mspot_meas vi, ret, pins()                                '13

ret = hp4156b_setSwitch(vi, hp4156b_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 =   0.1
vb =       0.7
ibcomp =   0.01
vc =       3
iccomp =   0.1

Dim mch(3) As Long                                                         '16
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                                                        '27
Dim st(2) As Long

ret = hp4156b_force(vi, pins(0), hp4156b_VF_MODE, 0, ve, iecomp, 0)        '30
ret = hp4156b_force(vi, pins(1), hp4156b_VF_MODE, 0, vb, ibcomp, 0)
ret = hp4156b_force(vi, pins(2), hp4156b_VF_MODE, 0, vc, iccomp, 0)
ret = hp4156b_measureM(vi, mch(0), mode(0), range(0), md(0), st(0))
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                               '36
display_data md(), st(), vi, ret, pins()
End Sub
```

| Line | Description |
|---|---|
| 1 | Beginning of the mspot_meas subprogram. |
| 3 to 28 | Declares variables used in the subprogram, and defines the value. |
| 30 to 35 | Applies voltage to device, and performs the multi channel spot measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 36 | Sets the specified port to the zero output state. |
| 37 | Calls the display_data subprogram (next page) to display measurement data. |
| 38 | End of the mspot_meas subprogram. |

```
Sub display_data(md() As Double, st() As Long, 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)"
   If st(1) = 0 Then
      value = value & Chr(10) & Chr(10) & "Ib = " & md(1) * 1000 & " (mA)"
      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                                                                         '25
End Sub
```

| Line | Description |
|---|---|
| 1 | Beginning of the display_data subprogram. |
| 3 to 6 | Declares variables, and defines the value. |
| 8 to 25 | 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.<br>Or displays error message on a message box if the status is abnormal. |
| 26 | End of the display_data subprogram. |

**Measurement Result Example**

```
Ic = 3.808 (mA)

Ib = 0.01883 (mA)

hfe = 202.230483271375

Do you want to perform measurement again?
```

# Staircase Sweep Measurements

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

**Table 3-4 Staircase Sweep 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) = 3          'SMU3: source
m(3) = 4          'SMU4: substrate

ret = hp4156b_setSwitch(vi, m(3), 1)                      '9
ret = hp4156b_setSwitch(vi, m(2), 1)
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret                                         '13

sweep_meas vi, ret, m()                                   '15

ret = hp4156b_setSwitch(vi, hp4156b_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 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 dvg(nop2) As Double                          '41

ret = hp4156b_force(vi, m(3), hp4156b_VF_MODE, 0, 0, 0.05, 0)
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, 0, 0.05, 0)
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 30 | Declares variables, and defines the value. |
| 33 to 41 | Declares variables used to keep source data, measurement data and status data. Also defines array size. |
| 43 to 44 | Applies voltage to device. |

```
Dim d_vg As Double        'secondary sweep step value (delta)                '46
If nop2 = 1 Then
    d_vg = 0
Else
    d_vg = (vg2 - vg1) / (nop2 - 1)
End If

Dim vg As Double          'secondary sweep source output
vg = vg1                                                                     '54

i = 0                     'array counter for sweepIv returned data           '56
For j = 1 To nop2         'array counter for secondary sweep output data
    dvg(j - 1) = vg
    ret = hp4156b_force(vi, m(1), hp4156b_VF_MODE, 0, vg, igcomp, 0)
    ret = hp4156b_setIv(vi, m(0), hp4156b_SWP_VF_SGLLIN, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
    ret = hp4156b_sweepIv(vi, m(0), hp4156b_IM_MODE, 0, rep, sc(i), md(i), st(i))
    check_err vi, ret
    vg = vg + d_vg
    If rep = nop1 Then                                                       '64
        i = i + nop1
    Else
        msg = rep & " measurement steps were returned. It must be " & nop1 & "
steps.   "
        MsgBox msg, vbOKOnly, ""
        ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)
        GoTo Bottom_sub
    End If
Next j                                                                       '72

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)
save_data nop1, nop2, dvg(), md(), st(), sc(), vi, ret, m()                  '75

Bottom_sub:
End Sub
```

| Line | Description |
|---|---|
| 46 to 54 | Declares variables, and defines the value. They are used for the secondary sweep source, d_vg is the step value and vg is the output value. |
| 56 to 72 | Performs the staircase sweep measurement (MOSFET Id-Vd characteristics). |
| 59 to 62 | Applies voltage to device, and sets the voltage sweep source. And performs the staircase sweep measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 64 to 71 | Displays a message box to notify error, disables the source outputs, and stops the program execution if the number of returned data is not equal to the nop1 value. |
| 75 | Calls the save_data subprogram (next page) to save measurement data. |
| 78 | 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, 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), 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 & "," & 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                                                            '25
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                                                                         '33

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). |
| 25 to 33 | 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. |
| 35 | End of the save_data subprogram. |

**Measurement Result Example**

```
Vg (V), Vd (V), Id (mA), Status
1,0,-0.00011721,0
1,0.3,3.1915,0
1,0.6,5.8795,0
1,0.9,8.1215,0
1,1.2,10.004,0
1,1.5,11.64,0
1,1.8,13.09,0
1,2.1,14.385,0
1,2.4,15.57,0
1,2.7,16.63,0
1,3,17.6,0
2,0,-0.000117215,0
2,0.3,4.178,0
2,0.6,7.9075,0
2,0.9,11.193,0
2,1.2,14.035,0
2,1.5,16.49,0
2,1.8,18.59,0
2,2.1,20.44,0
2,2.4,22.095,0
2,2.7,23.575,0
2,3,24.94,0
3,0,0.00050875,0
3,0.3,5.0385,0
3,0.6,9.6655,0
3,0.9,13.88,0
3,1.2,17.65,0
3,1.5,21.005,0
3,1.8,23.935,0
3,2.1,26.515,0
3,2.4,28.775,0
3,2.7,30.77,0
3,3,32.575,0

Data save completed.

Do you want to perform measurement again?
```

# Synchronous Sweep Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform 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.

**Table 3-5** **Synchronous Sweep 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) = 3          ’SMU3: source
m(3) = 4          ’SMU4: substrate

ret = hp4156b_setSwitch(vi, m(3), 1)                     ’9
ret = hp4156b_setSwitch(vi, m(2), 1)
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret                                        ’13

sweep_meas vi, ret, m()                                  ’15

ret = hp4156b_setSwitch(vi, hp4156b_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                                                      '29

Dim rep             As Long                                   '31
Dim sc()            As Double 'primary sweep output data
Dim md()            As Double 'sweep measurement data
Dim st()            As Long   'status data at each step

ReDim Preserve sc(nop) As Double
ReDim Preserve md(nop) As Double
ReDim Preserve st(nop) As Long                                '38
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 29 | Declares variables, and defines the value. |
| 31 to 38 | Declares variables used to keep source data, measurement data and status data. Also defines array size. |

```
ret = hp4156b_force(vi, m(3), hp4156b_VF_MODE, 0, vcon1, 0.05, 0)             '40
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, vcon2, 0.05, 0)
ret = hp4156b_setIv(vi, m(1), hp4156b_SWP_VF_SGLLIN, 0, vpri1, vpri2, nop, hold,
delay, s_delay, i1comp, p1comp)
check_err vi, ret
ret = hp4156b_setSweepSync(vi, m(0), hp4156b_VF_MODE, 0, vsyn1, vsyn2, i2comp,
P2comp)
check_err vi, ret

ret = hp4156b_sweepIv(vi, m(0), hp4156b_IM_MODE, 0, rep, sc(0), md(0), st(0)) '47
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                  '50

Dim msg As String
If rep = nop Then                                                             '53
    save_data nop, md(), st(), sc(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be " & nop & " steps.   "
    MsgBox msg, vbOKOnly, ""
End If                                                                        '58

End Sub                                                                       '60
```

| Line | Description |
|---|---|
| 40 to 41 | Applies voltage to device. |
| 42 | Sets the primary sweep source. |
| 44 | Sets the synchronous sweep source. |
| 47 | Performs the staircase sweep measurement. |
| 50 | Sets the specified port to the zero output state. |
| 43, 45, and 48 | Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 53 to 58 | 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. |
| 60 | End of the sweep_meas subprogram. |

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() 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), Status"

For i = 0 To nop - 1                                                         '7
   val = val & Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & 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                                                          '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()     'returns to sweep_meas if Yes is clicked.
End If                                                                       '30

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). |
| 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**

```
Vg (V), Id (mA), Status
0,-0.000098485,0
0.3,2.338,0
0.6,4.9295,0
0.9,7.7645,0
1.2,10.8095,0
1.5,14.05,0
1.8,17.465,0
2.1,21.045,0
2.4,24.755,0
2.7,28.59,0
3,32.54,0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Sweep Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform 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 3-6** **Multi-Channel Sweep Measurement Example**

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

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

ret = hp4156b_setSwitch(vi, m(2), 1)                      '8
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret

sweep_meas vi, ret, m()                                   '13

ret = hp4156b_setSwitch(vi, hp4156b_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
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
n =           nop *2

Dim mch(3)    As Long
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
Dim rep                  As Long
Dim sc()                 As Double  'primary sweep output data
Dim md()                 As Double  'sweep measurement data
Dim st()                 As Long    'status data at each step
ReDim Preserve sc(nop)   As Double
ReDim Preserve md(n)     As Double
ReDim Preserve st(n)     As Long                              '46
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 46 | Declares variables used in this subprogram, and defines the value. |

```
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2)                ’48
check_err vi, ret

ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, ve, iecomp, 0)              ’51
ret = hp4156b_force(vi, m(1), hp4156b_VF_MODE, 0, vc, iccomp, 0)
ret = hp4156b_setIv(vi, m(0), hp4156b_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp)
check_err vi, ret

ret = hp4156b_sweepMiv(vi, mch(0), mode(0), range(0), rep, sc(0), md(0), st(0))
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                  ’59

Dim msg As String
If rep = nop Then                                                             ’62
    save_data nop, md(), st(), sc(), vi, ret, m()
Else
    msg = rep & " measurement steps were returned. It must be " & nop & " steps."
    MsgBox msg, vbOKOnly, ""
End If                                                                        ’67

End Sub                                                                       ’69
```

| Line | Description |
|---|---|
| 48 | Sets the A/D converter integration time. |
| 51 to 52 | Applies voltage to device. |
| 53 | Sets the staircase sweep source. |
| 56 | Performs multi channel sweep measurement. |
| 59 | Sets the specified port to the zero output state. |
| 49, 54, and 57 | Calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 62 to 67 | 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. |
| 69 | End of the sweep_meas subprogram. |

```
Sub save_data(nop As Long, md() As Double, st() As Long, sc() 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 = "Vb (V), Ib (mA), Ic (mA), Status_b, Status_c"

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

Dim fname   As String           'data file name                              '12
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                   '18
Print #fnum, val
Close fnum

'displays data on a MsgBox
Dim title As String                                                          '23
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                                                                       '31

End Sub
```

| 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. |
| 12 to 20 | Saves measurement data into a file (C:\Agilent\data\data3.txt, 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**

```
Vb (V), Ib (mA), Ic (mA), Status_b, Status_c
0.3,0,-0.005,0,0
0.35,0,-0.005,0,0
0.4,0,-0.005,0,0
0.45,0,-0.005,0,0
0.5,0,0,0,0
0.55,0.0001,0.015,0,0
0.6,0.0005,0.085,0,0
0.65,0.00305,0.605,0,0
0.7,0.01915,3.89,0,0
0.75,0.09975,19.625,0,0
0.8,0.34745,59.38,0,0

Data save completed.

Do you want to perform measurement again?
```

# Pulsed Spot Measurements

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

**Table 3-7** **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) = 3          'SMU3: source
pins(3) = 4          'SMU4: substrate

ret = hp4156b_setSwitch(vi, pins(3), 1)                   '9
ret = hp4156b_setSwitch(vi, pins(2), 1)
ret = hp4156b_setSwitch(vi, pins(1), 1)
ret = hp4156b_setSwitch(vi, pins(0), 1)
check_err vi, ret                                         '13

spot_meas vi, ret, pins()                                 '15

ret = hp4156b_setSwitch(vi, hp4156b_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 base As Double
Dim width As Double
Dim period As Double
Dim hold As Double
vd = 0.5
idcomp = 0.05
vg = 0.5
igcomp = 0.01
base = 0
width = 0.001
period = 0.01
hold = 0.1
Dim meas As Double
Dim status As Long                                                              '20

ret = hp4156b_setFilter(vi, pins(1), hp4156b_FLAG_OFF)                          '22
ret = hp4156b_setPbias(vi, pins(1), hp4156b_VF_MODE, 2, base, vg, width, period,
hold, igcomp)
ret = hp4156b_force(vi, pins(3), hp4156b_VF_MODE, 0, 0, 0.05, 0)
ret = hp4156b_force(vi, pins(2), hp4156b_VF_MODE, 0, 0, 0.05, 0)
ret = hp4156b_force(vi, pins(0), hp4156b_VF_MODE, 2, vd, idcomp, 0)
ret = hp4156b_measureP(vi, pins(0), hp4156b_IM_MODE, 0, meas, status)           '27
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                    '30
display_data meas, status, vi, ret, pins()
End Sub
```

| Line | Description |
|---|---|
| 1 | Beginning of the spot_meas subprogram. |
| 3 to 20 | Declares variables used in this subprogram, and defines the value. |
| 22 to 23 | Sets the SMU filter off for the pulsed bias channel, and set the pulsed bias source. |
| 24 to 26 | Applies voltage to device. |
| 27 to 28 | Performs the pulsed spot measurement, and calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 30 | Sets the specified port to the zero output state. |
| 31 | Calls the display_data subprogram (next page) to display measurement data. |
| 32 | End of the spot_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                                                '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.<br><br>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?
```

# Multi-Channel Pulsed Spot Measurements

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

**Table 3-8 Multi-Channel Pulsed 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 = hp4156b_setSwitch(vi, pins(2), 1)                  ’8
ret = hp4156b_setSwitch(vi, pins(1), 1)
ret = hp4156b_setSwitch(vi, pins(0), 1)
check_err vi, ret                                        ’11

mspot_meas vi, ret, pins()                               ’13

ret = hp4156b_setSwitch(vi, hp4156b_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 pulsed 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 iccomp   As Double
Dim ibcomp   As Double
vc = 0
vb = 0
iccomp = 0.1
ibcomp = 0.1
Dim mch(3)   As Long
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 eod      As Long
Dim dtype    As Long
Dim mdata    As Double
Dim stat     As Long
Dim ch       As Long
Dim md(2)    As Double
Dim st(2)    As Long                                                                       '27

ret = hp4156b_setFilter(vi, pins(0), hp4156b_FLAG_OFF)                                     '29
ret = hp4156b_cmd(vi, "PT 0,0.005,0.01,0,1")                  'sets pulse timing parameters
ret = hp4156b_cmd(vi, "PV 1,0,0,-0.8,0.1")                    'sets pulse voltage source
ret = hp4156b_force(vi, pins(1), hp4156b_VF_MODE, 0, vb, ibcomp, 0)                        '32
ret = hp4156b_force(vi, pins(2), hp4156b_VF_MODE, 0, vc, iccomp, 0)
ret = hp4156b_startMeasure(vi, hp4156b_MM_PSPOT, mch(0), mode(0), range(0), hp4156b_FLAG_OFF)
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                               '37
ret = hp4156b_readData(vi, eod, dtype, mdata, stat, ch)
md(0) = mdata
st(0) = stat
ret = hp4156b_readData(vi, eod, dtype, mdata, stat, ch)
md(1) = mdata
st(1) = stat
display_data md(), st(), vi, ret, pins()
End Sub                                                                                    '45
```

| Line | Description |
|---|---|
| 1 | Beginning of the mspot_meas subprogram. |
| 3 to 27 | Declares variables used in this subprogram, and defines the value. |
| 29 to 31 | Sets the SMU filter off for the pulsed bias channel, and set the pulsed bias source. |
| 32 to 35 | Applies voltage to device, and performs multi channel pulsed spot measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 37 to 43 | Sets all channels to zero output state, and read the measurement result data. |
| 44 | Calls the display_data subprogram (next page) to display measurement data. |
| 45 | End of the mspot_meas subprogram. |

```
Sub display_data(md() As Double, st() 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"

If st(0) = 0 Then                                                                '8
  value = "Ic = " & md(0) * 1000 & " (mA)" & Chr(10) & Chr(10)
  If st(1) = 0 Then
    value = value & "Ib = " & md(1) * 1000 & " (mA)" & Chr(10) & Chr(10)
    value = value & "hfe = " & md(0)/md(1) & Chr(10) & Chr(10)
    value = value & "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
End Sub                                                                          '26
```

| Line | Description |
|---|---|
| 1 | Beginning of the display_data subprogram. |
| 3 to 6 | Declares variables, and defines the value. |
| 8 to 25 | 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.<br>Or displays error message on a message box if the status is abnormal. |
| 26 | End of the display_data subprogram. |

**Measurement Result Example**

```
Ic = 42.1918 (mA)

Ib = 0.321449 (mA)

hfe = 131.255035791059

Do you want to perform measurement again?
```

# Pulsed Sweep Measurements

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

**Table 3-9** **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 = hp4156b_setSwitch(vi, m(2), 1)                         '8
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret

sweep_meas vi, ret, m()                                      '13

ret = hp4156b_setSwitch(vi, hp4156b_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 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
nop1 = 11
nop2 = 3
Dim i As Integer
Dim j As Integer
Dim n As Long
n = nop1 * nop2
Dim rep             As Long
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 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 dib(nop2) As Double

ret = hp4156b_setFilter(vi, m(0), hp4156b_FLAG_OFF)                     '41
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2)
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, 0, 0.05, 0)
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 39 | Declares variables used in this subprogram, and defines the value. |
| 41 | Sets the SMU filter off for the pulse output channel. |
| 42 | Sets the A/D converter integration time. |
| 43 | Applies voltage to device. |

```
Dim d_ib As Double   'secondary sweep step value (delta)                    '45
If nop2 = 1 Then
    d_ib = 0
Else
    d_ib = (ib2 - ib1) / (nop2 - 1)
End If

Dim ibo As Double    'secondary sweep source output
ibo = ib1                                                                   '53

Dim msg As String
i = 0                'array counter for sweepIv returned data
For j = 1 To nop2
  dib(j - 1) = ibo
  ret = hp4156b_force(vi, m(1), hp4156b_IF_MODE, 0, ibo, vbcomp, 0)         '59
  ret = hp4156b_setPiv(vi, m(0), hp4156b_SWP_VF_SGLLIN, 0, base, vc1, vc2, nop1,
hold, width, period, iccomp)
  ret = hp4156b_sweepPiv(vi, m(0), hp4156b_IM_MODE, 0, rep, sc(i), md(i), st(i))
  check_err vi, ret
  ibo = ibo + d_ib
  If rep = nop1 Then                                                        '64
     i = i + nop1
  Else
     msg = rep & " measurement steps were returned. It must be " & nop1 & " steps."
     MsgBox msg, vbOKOnly, ""
     ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)
     GoTo Bottom_sub
  End If
Next j                                                                      '72

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                '74
save_data nop1, nop2, md(), st(), sc(), dib(), vi, ret, m()
Bottom_sub:
End Sub                                                                     '77
```

| Line | Description |
|---|---|
| 45 to 53 | Declares variables, and defines the value. They are used for the secondary sweep source, d_ib is the step value and ibo is the output value. |
| 59 to 62 | Applies current to device, sets the pulsed sweep voltage source, and performs the pulsed sweep measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 64 to 71 | Disables the channels and stops the program execution if the number of returned data is not equal to the nop1 value. |
| 74 | Sets the specified port to the zero output state. |
| 75 | Calls the save_data subprogram (next page) to save measurement data. |
| 77 | End of the sweep_meas subprogram. |

```
Sub save_data(nop1 As Long, nop2 As Long, md() As Double, st() As Long, sc() 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), Status"

For j = 1 To nop2                                                              '8
  For i = nop1 * (j - 1) To nop1 * j - 1
     val = val & Chr(13) & Chr(10) & dib(j - 1) * 1000000# & "," & sc(i) & "," &
md(i) * 1000 & "," & st(i)
  Next i
Next j

Dim fname   As String      'data file name                                     '14
Dim fnum    As Integer     'file number
fname = "C:\Agilent\data\data4.txt"
fnum = 1
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum

Dim title   As String                                                          '22
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()
End If

End Sub                                                                        '32
```

| 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 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**

```
Ib (uA), Vc (V), Ic (mA), Status
50,0,-0.05,0
50,0.3,8.965,0
50,0.6,9.705,0
50,0.9,9.735,0
50,1.2,9.765,0
50,1.5,9.805,0
50,1.8,9.83,0
50,2.1,9.835,0
50,2.4,9.85,0
50,2.7,9.9,0
50,3,9.915,0
100,0,-0.1,0
100,0.3,15.725,0
100,0.6,18.115,0
100,0.9,18.715,0
100,1.2,18.84,0
100,1.5,18.925,0
100,1.8,19.015,0
100,2.1,19.045,0
100,2.4,19.12,0
100,2.7,19.175,0
100,3,19.215,0
150,0,-0.15,0
150,0.3,21.065,0
150,0.6,24.54,0
150,0.9,26.47,0
150,1.2,27.19,0
150,1.5,27.405,0
150,1.8,27.605,0
150,2.1,27.71,0
150,2.4,27.795,0
150,2.7,27.885,0
150,3,27.955,0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Pulsed Sweep Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform multi channel pulsed 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 3-10 Multi-Channel 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) = 1          'SMU1: emitter
m(1) = 2          'SMU2: base
m(2) = 4          'SMU4: collector

ret = hp4156b_setSwitch(vi, m(0), 1)                      '8
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(2), 1)
check_err vi, ret

sweep_meas vi, ret, m()                                   '13

ret = hp4156b_setSwitch(vi, hp4156b_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 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 vc       As Double                                      '3
Dim vb       As Double
Dim iccomp   As Double
Dim ibcomp   As Double
Dim nop      As Long
vc =         0
vb =         0
iccomp =     0.1
ibcomp =     0.1
nop =        11
Dim mch(3)   As Long
Dim mode(2)  As Long
Dim range(2) As Double
mch(0) =     m(2)        'measurement channel: collector
mch(1) =     m(1)        'measurement channel: base
mch(2) =     0
mode(0) =    1           'current measurement mode
mode(1) =    1           'current measurement mode
range(0) =   0           'auto range
range(1) =   0           'auto range
Dim rep                 As Long
Dim sc()                As Double  'primary sweep output data
Dim md1()               As Double  'sweep measurement data
Dim md2()               As Double  'sweep measurement data
ReDim Preserve sc(nop)  As Double
ReDim Preserve md1(nop) As Double
ReDim Preserve md2(nop) As Double
Dim eod      As Long
Dim dtype    As Long
Dim mdata    As Double
Dim stat     As Long
Dim ch       As Long                                        '34
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 34 | Declares variables used in this subprogram, and defines the value. |

```
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2)                ’36
ret = hp4156b_setFilter(vi, m(0), hp4156b_FLAG_OFF)
ret = hp4156b_cmd(vi, "PT 0,0.005,0.01,0,1")          ’sets pulse timing parameters
ret = hp4156b_cmd(vi, "PWV 1,1,0,0,0,-0.8,11,0.1,0") ’sets pulsed sweep source
check_err vi, ret                                                             ’40

ret = hp4156b_force(vi, m(1), hp4156b_VF_MODE, 0, vb, ibcomp, 0)              ’42
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, vc, iccomp, 0)
ret = hp4156b_startMeasure(vi, hp4156b_MM_PSWEEP, mch(0), mode(0), range(0),
hp4156b_FLAG_ON)
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                  ’47
Dim i As Integer
For i = 1 To nop
  ret = hp4156b_readData(vi, eod, dtype, mdata, stat, ch)
  md1(i) = mdata ’ Data measured by mch(0) = collector
  ret = hp4156b_readData(vi, eod, dtype, mdata, stat, ch)
  md2(i) = mdata ’ Data measured by mch(1) = base
  ret = hp4156b_readData(vi, eod, dtype, mdata, stat, ch)
  sc(i) = mdata ’ Pulsed sweep source output data
Next i

save_data nop, m(), sc(), md1(), md2(), vi, ret
End Sub                                                                       ’59
```

| Line | Description |
|---|---|
| 36 to 40 | Sets the A/D converter integration time, sets the SMU filter off for the pulsed sweep source, sets the pulse timing parameters, and sets the pulsed sweep source. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 42 to 45 | Applies voltage to device, and performs multi channel pulsed sweep measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 47 to 56 | Sets all channels to zero output state, and reads the measurement result data. |
| 58 | Calls the save_data subprogram to save measurement data. |
| 59 | End of the sweep_meas subprogram. |

```
Sub save_data(nop As Long, m() As Long, sc() As Double, md1() As Double, md2() As Double,
vi As Long, ret As Long)                                                          '1
Dim i     As Integer       'array counter for primary sweep                       '2
Dim val   As String        'data to be saved to a file
val = "Ve (V), Ic (mA), Ib (mA)"
For i = 1 To nop
  val = val & Chr(13) & Chr(10) & sc(i) & "," & md1(i) * 1000 & "," & md2(i) * 1000
Next i
Dim fname   As String           'data file name                                   '8
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data5.txt"
fnum = 1
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum
Dim title As String                                                               '15
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()
End If
End Sub                                                                           '24
```

| Line | Description |
|---|---|
| 1 | Beginning of the save_data subprogram. |
| 2 to 7 | Declares variables, and creates data to be saved and displayed on a message box. |
| 8 to 14 | Saves measurement data into a file (C:\Agilent\data\data5.txt, CSV file). |
| 15 to 23 | 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. |
| 24 | End of the save_data subprogram. |

**Measurement Result Example**

```
Vb (V), Ib (mA), Ic (mA), Status_b, Status_c
0.3,0,-0.005,0,0
0.35,0,-0.005,0,0
0.4,0,-0.005,0,0
0.45,0,-0.005,0,0
0.5,0,0,0,0
0.55,0.0001,0.015,0,0
0.6,0.0005,0.085,0,0
0.65,0.00305,0.605,0,0
0.7,0.01915,3.89,0,0
0.75,0.09975,19.625,0,0
0.8,0.34745,59.38,0,0

Data save completed.

Do you want to perform measurement again?
```

# Staircase Sweep with Pulsed Bias Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform 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) = 3          'SMU3: source
m(3) = 4          'SMU4: substrate

ret = hp4156b_setSwitch(vi, m(3), 1)                      '9
ret = hp4156b_setSwitch(vi, m(2), 1)
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret

sweep_meas vi, ret, m()                                   '15

ret = hp4156b_setSwitch(vi, hp4156b_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 with pulsed bias 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                                                            '27

Dim i As Integer
Dim j As Integer
Dim n As Long
n = nop1 * nop2

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 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 dvg(nop2) As Double                                  '43

ret = hp4156b_setFilter(vi, m(1), hp4156b_FLAG_OFF)                 '45
ret = hp4156b_force(vi, m(3), hp4156b_VF_MODE, 0, 0, 0.05, 0)
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, 0, 0.05, 0)
```

| Line | Description |
|---|---|
| 1 | Beginning of the sweep_meas subprogram. |
| 3 to 32 | Declares variables for source channels, and defines the value. |
| 35 to 43 | Declares variables used to keep source data, measurement data and status data. Also defines array size. |
| 45 to 47 | Sets the SMU filter off for the pulsed bias channel, and applies voltage to device. |

```
Dim d_vg As Double 'secondary sweep step value (delta)                                    '49
If nop2 = 1 Then
    d_vg = 0
Else
    d_vg = (vg2 - vg1) / (nop2 - 1)
End If
Dim vg As Double   'secondary sweep source output
vg = vg1

Dim width       As Double                                                                 '58
Dim period      As Double
Dim p_hold      As Double
width = 0.001
period = 0.01
p_hold = 0.1
i = 0                       'array counter for sweepIv returned data
For j = 1 To nop2           'array counter for secondary sweep output data               '65
    dvg(j - 1) = vg
    ret = hp4156b_setPbias(vi, m(1), hp4156b_VF_MODE, 0, 0, vg, width, period, p_hold, igcomp)
    ret = hp4156b_setIv(vi, m(0), hp4156b_SWP_VF_SGLLIN, 0, vd1, vd2, nop1, hold, delay,
s_delay, idcomp, p_comp)
    ret = hp4156b_sweepPbias(vi, m(0), hp4156b_IM_MODE, 0, rep, sc(i), md(i), st(i))
    check_err vi, ret

    vg = vg + d_vg
    If rep = nop1 Then                                                                    '73
        i = i + nop1
    Else
        msg = rep & " measurement steps were returned. It must be " & nop1 & " steps."
        MsgBox msg, vbOKOnly, ""
        ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)
        GoTo Bottom_sub
    End If                                                                                '80
Next j
ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)
save_data nop1, nop2, dvg(), md(), st(), sc(), vi, ret, m()
Bottom_sub:
End Sub                                                                                   '85
```

| Line | Description |
|---|---|
| 49 to 56 | Declares variables, and defines the value. They are used for the secondary sweep source, d_vg is the step value and vg is the output value. |
| 67 to 70 | Sets the pulsed source, sets the voltage sweep source, and performs the staircase sweep with pulsed bias measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 73 to 80 | Disables the channels and stops the program execution if the number of returned data is not equal to the nop1 value. |
| 82 to 83 | Sets all channels to zero output state, and calls the save_data subprogram (next page) to save measurement data. |
| 85 | 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, vi As Long, ret As Long, m() As Long)                    '1

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

Dim fname   As String          'data file name                                 '13
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                                                            '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()
End If

End Sub                                                                        '31
```

| Line | Description |
|---|---|
| 1 | Beginning of the save_data subprogram. |
| 3 to 11 | Declares variables, and creates data to be saved and displayed on a message box. |
| 13 to 19 | Saves measurement data into a file (C:\Agilent\data\data6.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), Vd (V), Id (mA), Status
1,0,0,0
1,0.3,3.205,0
1,0.6,5.9,0
1,0.9,8.15,0
1,1.2,10.035,0
1,1.5,11.68,0
1,1.8,13.13,0
1,2.1,14.425,0
1,2.4,15.61,0
1,2.7,16.675,0
1,3,17.65,0
2,0,-0.005,0
2,0.3,4.205,0
2,0.6,7.955,0
2,0.9,11.245,0
2,1.2,14.11,0
2,1.5,16.55,0
2,1.8,18.67,0
2,2.1,20.52,0
2,2.4,22.185,0
2,2.7,23.67,0
2,3,25.02,0
3,0,0,0
3,0.3,5.07,0
3,0.6,9.73,0
3,0.9,13.965,0
3,1.2,17.76,0
3,1.5,21.115,0
3,1.8,24.07,0
3,2.1,26.64,0
3,2.4,28.91,0
3,2.7,30.925,0
3,3,32.71,0

Data save completed.

Do you want to perform measurement again?
```

# Sampling Measurements

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform sampling measurement (sampling_meas), and save measurement result data into a file (save_data). This example measures current of a device that has two high terminals and a low terminal, and calculates the resistance.

**Table 3-12 Sampling Measurement Example**

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

Dim m(3) As Long  ’SMU port numbers                      ’3
m(0) = 1          ’SMU1: t1
m(1) = 2          ’SMU2: t2
m(2) = 3          ’SMU3: low

ret = hp4156b_setSwitch(vi, m(2), 1)                     ’8
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret

sampling_meas vi, ret, m()                               ’13

ret = hp4156b_setSwitch(vi, hp4156b_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 sampling_meas subprogram (next page) to perform sampling 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 sampling_meas(vi As Long, ret As Long, m() As Long)      '1

Dim base     As Double                                       '3
Dim bias     As Double
Dim icomp    As Double
Dim vlout    As Double
Dim ilcomp   As Double
Dim hold     As Double
Dim interval As Double
Dim nop      As Long
base =       0
bias =       0.1
icomp =      0.1
vlout =      0
ilcomp =     0.1
hold =       0.1
interval =   0.05
nop =        30

Dim mch(3)   As Long
Dim mode(2)  As Long
Dim range(2) As Double
Dim n        As Long
n = nop * 2
mch(0) =   m(0)   ' t1
mch(1) =   m(1)   ' t2
mch(2) =   0
mode(0) =  1
mode(1) =  1
range(0) = 0
range(1) = 0

Dim point    As Long                                         '33
Dim index()  As Long
Dim value()  As Double
Dim status() As Long
ReDim Preserve index(nop) As Long
ReDim Preserve value(n)   As Double
ReDim Preserve status(n)  As Long                            '39
```

| Line | Description |
|---|---|
| 1 | Beginning of the sampling_meas subprogram. |
| 3 to 31 | Declares variables, and defines the value. |
| 33 to 39 | Declares variables used to keep index data, measurement data, and status data. Also defines array size. |

```
ret = hp4156b_setFilter(vi, hp4156b_CH_ALL, hp4156b_FLAG_ON)                    '41
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2)

ret = hp4156b_setSample(vi, hold, interval, nop)                                '44
ret = hp4156b_addSampleSyncIv(vi, m(0), hp4156b_VF_MODE, 0, base, bias, icomp)
ret = hp4156b_addSampleSyncIv(vi, m(1), hp4156b_VF_MODE, 0, base, bias, icomp)
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, 0, vlout, ilcomp, 0)

ret = hp4156b_sample(vi, mch(0), mode(0), range(0), point, index(0), value(0),
status(0))                                                                      '49
check_err vi, ret

ret = hp4156b_clearSampleSync(vi)                                               '52
ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)

Dim msg As String                                                               '55
If point <> nop Then
  msg = point & " measurement steps were returned. It must be " & nop & " steps."
  MsgBox msg, vbOKOnly, ""
Else
  save_data vi, ret, nop, index(), bias, value(), status(), m()
End If

End Sub                                                                         '63
```

| Line | Description |
|---|---|
| 41 to 42 | Sets the SMU filter on for the all channels, and sets the A/D converter integration time. |
| 44 to 47 | Sets the sampling timing parameters, sets the voltage sources that operate synchronously with the measurement trigger, and applies voltage to device. |
| 49 to 50 | Performs the sampling measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 52 to 53 | Clears the setting of the voltage sources defined by the hp4156b_addSampleSyncIv function, and sets all channels to zero output state. |
| 55 to 61 | Calls the save_data subprogram (next page) to save the measurement result data. However, if the number of returned data is not equal to the nop value, the program disables the source/measurement channels and stops program execution without saving the measurement data. |
| 63 | End of the sampling_meas subprogram. |

```
Sub save_data(vi As Long, ret As Long, nop As Long, index() As Long, bias As
Double, value() As Double, status() As Long, m() As Long)                    '1

Dim i     As Integer                                                         '3
Dim data1 As Double
Dim data2 As Double
Dim r1    As Double
Dim r2    As Double
Dim val   As String

val = "Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status"
For i = 0 To nop - 1
  data1 = Round(value(2 * i) * 1000, 3)
  data2 = Round(value(2 * i + 1) * 1000, 3)
  r1 = Round(bias / value(2 * i), 2)
  r2 = Round(bias / value(2 * i + 1), 2)
  val = val & Chr(13) & Chr(10) & index(i) & "," & data1 & "," & r1
  val = val & "," & data2 & "," & r2 & "," & status(i)
Next i

Dim fname   As String           'data file name                              '20
Dim fnum    As Integer          'file number
fname = "C:\Agilent\data\data7.txt"
fnum = 1
Open fname For Output Access Write Lock Read Write As fnum
Print #fnum, val
Close fnum

Dim title As String                                                          '28
Dim rbx   As Integer
title = "Sampling 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
    sampling_meas vi, ret, m()
End If

End Sub                                                                      '38
```

| Line | Description |
|---|---|
| 1 | Beginning of the save_data subprogram. |
| 3 to 18 | Declares variables, and creates data to be saved and displayed on a message box. |
| 20 to 26 | Saves measurement data into a file (C:\Agilent\data\data7.txt, CSV file). |
| 28 to 36 | Displays measurement data on a message box. If Yes is clicked on the message box, performs the sampling_meas subprogram again. If No is clicked, returns to the perform_meas subprogram. |
| 38 | End of the save_data subprogram. |

**Measurement Result Example**

```
Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status
1,11.136,8.98,10.755,9.29,0
2,11.136,8.98,10.759,9.29,0
3,11.147,8.97,10.766,9.28,0
4,11.136,8.98,10.762,9.29,0
5,11.136,8.98,10.759,9.29,0
6,11.147,8.97,10.762,9.29,0
7,11.143,8.97,10.755,9.29,0
8,11.143,8.97,10.769,9.28,0
9,11.136,8.98,10.752,9.31,0
10,11.133,8.98,10.759,9.29,0
11,11.147,8.97,10.752,9.31,0
12,11.136,8.98,10.759,9.29,0
13,11.136,8.98,10.755,9.29,0
14,11.147,8.97,10.766,9.28,0
15,11.141,8.97,10.755,9.29,0
16,11.143,8.97,10.762,9.29,0
17,11.153,8.96,10.762,9.29,0
18,11.147,8.97,10.759,9.29,0
19,11.143,8.97,10.755,9.29,0
20,11.143,8.97,10.755,9.29,0
21,11.141,8.97,10.762,9.29,0
22,11.141,8.97,10.755,9.29,0
23,11.141,8.97,10.759,9.29,0
24,11.136,8.98,10.759,9.29,0
25,11.133,8.98,10.749,9.31,0
26,11.136,8.98,10.762,9.29,0
27,11.136,8.98,10.755,9.29,0
28,11.136,8.98,10.766,9.28,0
29,11.133,8.98,10.755,9.29,0
30,11.147,8.97,10.759,9.29,0

Data save completed.

Do you want to perform measurement again?
```

# Stress Force

This section explains example subprograms that enable/disable measurement channels (perform_meas), perform stress measurement (stress_meas), and display measurement result data (disp_data1 and disp_data2). This example measures current of a device that has two high terminals and a low terminal.

**Table 3-13** **Stress Force Example**

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

Dim m(3) As Long  'SMU port numbers                       '3
m(0) = 1          'SMU1: t1
m(1) = 2          'SMU2: t2
m(2) = 3          'SMU3: low

ret = hp4156b_setSwitch(vi, m(2), 1)                      '8
ret = hp4156b_setSwitch(vi, m(1), 1)
ret = hp4156b_setSwitch(vi, m(0), 1)
check_err vi, ret

stress_meas vi, ret, m()                                  '13

ret = hp4156b_setSwitch(vi, hp4156b_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 stress_meas subprogram (next page) to perform stress 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 stress_meas(vi As Long, ret As Long, m() As Long)              '1

Dim range    As Double                                             '3
Dim base     As Double
Dim stress   As Double
Dim bias     As Double
Dim icomp    As Double
Dim vlout    As Double
Dim ilcomp   As Double
Dim hold     As Double
Dim duration As Double
Dim period   As Long
Dim status   As Long
Dim md(4)    As Double
Dim st(4)    As Long

range =    0
base =     0
stress =   2
bias =     0.1
icomp =    0.1
vlout =    0
ilcomp =   0.1
hold =     0
duration = 5
period =   0.01

ret = hp4156b_setFilter(vi, hp4156b_CH_ALL, hp4156b_FLAG_ON)       '28
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2)
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, range, vlout, ilcomp, 0)
ret = hp4156b_force(vi, m(1), hp4156b_VF_MODE, range, bias, icomp, 0)
ret = hp4156b_force(vi, m(0), hp4156b_VF_MODE, range, bias, icomp, 0)
ret = hp4156b_spotMeas(vi, m(0), hp4156b_IM_MODE, range, md(0), st(0))
ret = hp4156b_spotMeas(vi, m(1), hp4156b_IM_MODE, range, md(1), st(1))
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                       '37
disp_data1 vi, ret, m(), md(), st(), stress, duration
```

| Line | Description |
|---|---|
| 1 | Beginning of the stress_meas subprogram. |
| 3 to 26 | Declares variables, and defines the value. |
| 28 to 35 | Sets the SMU filter on for the all channels, sets the A/D converter integration time, applies voltage to device, and performs spot measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 37 to 38 | Sets all channels to zero output state, and calls the disp_data1 subprogram to display measurement data. |

```
ret = hp4156b_setStress(vi, hold, hp4156b_STT_TIME, duration, period)          '40
ret = hp4156b_force(vi, m(2), hp4156b_VF_MODE, range, vlout, ilcomp, 0)
ret = hp4156b_addStressSyncIv(vi, 1, m(0), 2, range, base, stress, icomp)
ret = hp4156b_addStressSyncIv(vi, 2, m(1), 2, range, base, stress, icomp)
ret = hp4156b_stress(vi, status)
check_err vi, ret

ret = hp4156b_clearStressSync(vi)                                              '47

ret = hp4156b_force(vi, m(1), hp4156b_VF_MODE, range, bias, icomp, 0)          '49
ret = hp4156b_force(vi, m(0), hp4156b_VF_MODE, range, bias, icomp, 0)
ret = hp4156b_spotMeas(vi, m(0), hp4156b_IM_MODE, range, md(2), st(2))
ret = hp4156b_spotMeas(vi, m(1), hp4156b_IM_MODE, range, md(3), st(3))
check_err vi, ret

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL)                                   '55
disp_data2 vi, ret, m(), md(), st(), stress, duration

End Sub                                                                        '58
```

| Line | Description |
|---|---|
| 40 to 45 | Sets the stress timing parameters, applies voltage to the device, sets the stress sources that operate synchronously with the stress trigger, and applies the stress to the device. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 47 | Clears the setting of the stress sources defined by the hp4156b_addStressSyncIv function. |
| 49 to 53 | Applies voltage to the device, and performs spot measurement. After that, calls the check_err subprogram (shown in Table 3-1) to check if an error status is returned for the previous line. |
| 55 to 56 | Sets all channels to zero output state, and calls the disp_data2 subprogram to display measurement data. |
| 58 | End of the stress_meas subprogram. |

```
Sub disp_data1(vi As Long, ret As Long, m() As Long, md() As Double, st() As Long,
stress As Double, duration As Double)                                          '1

Dim title As String                                                            '3
Dim val As String
Dim rbx As Integer
title = "Measurement Result"
val = "Data before stress:"
val = val & Chr(13) & Chr(10) & "T1 current = " & md(0) * 1000 & " [mA]"
val = val & ", Status = " & st(0)
val = val & Chr(13) & Chr(10) & "T2 current = " & md(1) * 1000 & " [mA]"
val = val & ", Status = " & st(1)
val = val & Chr(10) & Chr(10) & "Do you want to CANCEL stress force?"
val = val & Chr(10) & Chr(10) & "Stress: " & stress & " [V], Duration: " & duration
& " [sec]"

Dim msg As String                                                              '15
rbx = MsgBox(val, vbYesNo, title)
If rbx = vbYes Then
  ret = hp4156b_close(vi)
  msg = "Click OK to stop the program."
  MsgBox msg, vbOKOnly, ""
  End
End If

End Sub                                                                        '24
```

| Line | Description |
|---|---|
| 1 | Beginning of the disp_data1 subprogram. |
| 3 to 13 | Declares variables, and creates data to be displayed on a message box. |
| 15 to 22 | Displays the message box. If Yes is clicked, stops the program execution. |
| 24 | End of the disp_data1 subprogram. |

```
Sub disp_data2(vi As Long, ret As Long, m() As Long, md() As Double, st() As Long,
stress As Double, duration As Double)                                          '1

Dim title As String                                                            '3
Dim val As String
Dim rbx As Integer
title = "Measurement Result"
val = "Data before stress:"
val = val & Chr(13) & Chr(10) & "T1 current = " & md(0) * 1000 & " [mA]"
val = val & ", Status = " & st(0)
val = val & Chr(13) & Chr(10) & "T2 current = " & md(1) * 1000 & " [mA]"
val = val & ", Status = " & st(1)
val = val & Chr(10) & Chr(10) & "Stress: " & stress & " [V], Duration: " & duration
& " [sec]"
val = val & Chr(10) & Chr(10) & "Data after stress:"
val = val & Chr(13) & Chr(10) & "T1 current = " & md(2) * 1000 & " [mA]"
val = val & ", Status = " & st(2)
val = val & Chr(13) & Chr(10) & "T2 current = " & md(3) * 1000 & " [mA]"
val = val & ", Status = " & st(3)
val = val & Chr(10) & Chr(10) & "Do you want to perform measurement AGAIN?"

rbx = MsgBox(val, vbYesNo, title)                                              '20
If rbx = vbYes Then
  stress_meas vi, ret, m()
End If

End Sub                                                                        '25
```

| Line | Description |
|---|---|
| 1 | Beginning of the disp_data2 subprogram. |
| 3 to 18 | Declares variables, and creates data to be displayed on a message box. |
| 20 to 23 | Displays the message box. If Yes is clicked, calls the stress_meas subprogram. |
| 25 | End of the disp_data2 subprogram. |

**Measurement Result Example**

```
Data before stress:
T1 current = 11.09860 [mA], Status = 0
T2 current = 10.75880 [mA], Status = 0

Stress: 2 [V], Duration: 5 [sec]

Data after stress:
T1 current = 11.10200 [mA], Status = 0
T2 current = 10.76220 [mA], Status = 0

Do you want to perform measurement AGAIN?
```

# 4 Programming Examples for Visual Basic .NET Users

This chapter describes how to create measurement programs using the Agilent 4155/4156 and the 4155/4156 VXI*plug&play* driver, and provides programming examples using Microsoft Visual Basic .NET. This chapter contains the following sections:

- “Programming Basics”
- “High-Speed Spot Measurements”
- “Multi-Channel Spot Measurements”
- “Staircase Sweep Measurements”
- “Synchronous Sweep Measurements”
- “Multi-Channel Sweep Measurements”
- “Pulsed Spot Measurements”
- “Multi-Channel Pulsed Spot Measurements”
- “Pulsed Sweep Measurements”
- “Multi-Channel Pulsed Sweep Measurements”
- “Staircase Sweep with Pulsed Bias Measurements”
- “Sampling Measurements”
- “Stress Force”

**NOTE**

## About Program Code

Programming examples are provided as subprograms that can be run with the project template shown in Table 4-1. To execute the program, insert the subprograms 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, hp4156b_reset(vi) must be entered as Ag415x.Reset() if you declare your 4155/4156 as Ag415x.

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. `Ag415x._`) 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. `Ag415x.Force(_ )`.

# Programming Basics

This section provides the basic information for programming using the Agilent 4155/4156 VXI*plug&play* driver.

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

**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.

## To Create Your Project Template

This section explains how to create a project template 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 4155/4156) 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. HP4156B - VXIplug&play).

**Step 5.** Right-click on the instrument icon (e.g. HP 4156C (: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. HP4156C).

**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 4155/4156 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. Ag415x).

**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 Ag415x As Hp4156b = New Hp4156b("GPIB0::17::INSTR", True, True)
    Ag415x.Reset()
    Ag415x.TimeOut(60000)

    MsgBox("Click OK to start measurement.", vbOKOnly, "")
    Console.WriteLine("Measurement in progress. . ." & Chr(10))
    perform_meas(Ag415x)

    Ag415x.SetSwitch(0, 0)
    Ag415x.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 4155/4156 VXI*plug&play* driver. |
| 11 to 24 | Main subprogram establishes the software connection with the Agilent 4155/4156, resets the 4155/4156, 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.<br><br>After the measurement, the program disables all channels, disables the software connection with the 4155/4156, 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 4155/4156 on the GPIB address 17. Confirm the GPIB address of your 4155/4156, and set the address correctly instead of “17”. |

```
  Sub perform_meas(ByVal Ag415x As Hp4156b)                                  ’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

    Ag415x.SetSwitch(1, 1)        ’this is dummy.                          ’38
    ’ insert measurement program code

    Ag415x.ErrorQuery(err, msg)                                            ’41
    If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

    Ag415x.ZeroOutput(0)                                                   ’44
    save_data(fname, title, val, data, nop1, nop2, Ag415x)

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.<br><br>*i* and *j*: Variables used to specify the element of the *data* array.<br>*nop1* and *nop2*: Number of measurement steps. Also used to declare the *data* array.<br>*data*: String data array used to store the measurement result data except for the header.<br>*val*: String data variable to store the header (first line) of the measurement result data.<br>*fname*: Full path name of the measurement result data file to be saved.<br>*title*: Title of the message box used to display the measurement result data.<br>*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 4155/4156 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 Ag415x
As Hp4156b)                                                                    '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(Ag415x)
  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 Measurements

This section explains an 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 Ag415x As Hp4156b)                                    '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\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
 Ag415x.SetSwitch(t(3), 1)        'SMU4: substrate
 Ag415x.SetSwitch(t(2), 1)        'SMU3: source
 Ag415x.SetSwitch(t(1), 1)        'SMU2: gate
 Ag415x.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

 Ag415x.Force(t(3), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A           26
 Ag415x.Force(t(2), 2, 0, 0, 0.1, 0)        'out= 0 V,  comp= 0.1 A
 Ag415x.Force(t(1), 2, 2, vg, igcomp, 0)    'out= vg V, comp= igcomp A
 Ag415x.Force(t(0), 2, 2, vd, idcomp, 0)    'out= vd V, comp= idcomp A
 Ag415x.ErrorQuery(err, msg)
 If err <> 0 Then Ag415x.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 24 | 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. |

```
 Ag415x.SpotMeas(t(0), 1, 0, meas, status)         'current measurement          33
 data(j, i) = Chr(13) & Chr(10) & meas * 1000 & ", " & status

 Ag415x.ZeroOutput(0)                                                            '36
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                       '39
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 33 to 34 | Performs spot measurement. And stores the measured data into the *data* valiable. |
| 36 to 37 | 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. |
| 39 to 40 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Id (mA), Status
3.91571, 0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Spot Measurements

This section explains an example subprogram that performs multi channel spot measurement. This example measures bipolar transistor collector current and base current.

**Table 4-3 Multi-Channel Spot Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), Status_c, Ib (mA), 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
 Ag415x.SetSwitch(t(2), 1)        'SMU3: collector
 Ag415x.SetSwitch(t(1), 1)        'SMU2: base
 Ag415x.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

 Ag415x.Force(t(0), Hp4156b.ModeEnum.VoltageOutput, 0, ve, iecomp, 0)        '30
 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, vb, ibcomp, 0)
 Ag415x.Force(t(2), Hp4156b.ModeEnum.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 28 | Declares variables and sets the value. |
| 30 to 32 | Applies voltage to device. |

```
 Ag415x.ErrorQuery(err, msg)                                                    '34
 If err <> 0 Then Ag415x.ZeroOutput(0): GoTo Check_err

 Ag415x.MeasureM(mch, mode, range, md, st)                                      '37
 data(j, i) = Chr(13) & Chr(10) & md(0) * 1000 & ", " & st(0) & ", "
 data(j, i) = data(j, i) & md(1) * 1000 & ", " & st(1) & ", " & md(0) / md(1)

 Ag415x.ZeroOutput(0)                                                           '41
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '44
 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 39 | Performs multi channel spot measurement. And stores the measured data into the *data* valiable. |
| 41 to 42 | 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. |
| 44 to 45 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Ic (mA), Status_c, Ib (mA), Status_b, hfe
3.79141, 0, 0.0187544, 0, 202.161092863541

Data save completed.

Do you want to perform measurement again?
```

# Staircase Sweep Measurements

This section explains an example subprogram that performs staircase sweep measurement. This example measures MOSFET Id-Vd characteristics.

**Table 4-4 Staircase Sweep Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), Status"
 Dim fname As String = "C:\Agilent\data\data3.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
 Ag415x.SetSwitch(t(3), 1)           'SMU4: substrate
 Ag415x.SetSwitch(t(2), 1)           'SMU3: source
 Ag415x.SetSwitch(t(1), 1)           'SMU2: gate
 Ag415x.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               '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. |

```
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)              '38
 Ag415x.Force(t(3), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)

 For j = 0 To nop2 - 1                                                          '41
  Ag415x.SetIv(t(0), Hp4156b.ModeEnum4.SingleLinearV, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
  Ag415x.ErrorQuery(err, msg)
  If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, vg, igcomp, 0)          '46
  Ag415x.SweepIv(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, 0, rep, sc, md, st)
  If rep <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                         '50
   data(j, i) = Chr(13) & Chr(10) & vg & "," & sc(i) & "," & md(i) * 1000 & "," &
st(i)
  Next i

  vg = vg + d_vg
 Next j

 Ag415x.ZeroOutput(0)                                                           '57
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '60
 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 48 | Applies voltage to the device 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. |
| 50 to 52 | Stores the measured data into the *data* valiable. |
| 57 to 58 | 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. |
| 60 to 62 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Vg (V), Vd (V), Id (mA), Status
1,0,-0.00115744,0
1,0.3,3.05372,0
1,0.6,5.61719,0
1,0.9,7.74477,0
1,1.2,9.53146,0
1,1.5,11.07517,0
1,1.8,12.4335,0
1,2.1,13.654,0
1,2.4,14.7489,0
1,2.699,15.7364,0
1,3,16.6246,0
2,0,-0.00174059,0
2,0.3,4.03733,0
2,0.6,7.63927,0
2,0.9,10.80673,0
2,1.2,13.5329,0
2,1.5,15.8655,0
2,1.8,17.8811,0
2,2.1,19.642,0
2,2.4,21.2001,0
2,2.699,22.6005,0
2,3,23.8623,0
3,0,-0.00185628,0
3,0.3,4.89763,0
3,0.6,9.39135,0
3,0.9,13.4735,0
3,1.2,17.1346,0
3,1.5,20.3616,0
3,1.8,23.1928,0
3,2.1,25.6613,0
3,2.4,27.8118,0
3,2.699,29.7103,0
3,3,31.4002,0

Data save completed.

Do you want to perform measurement again?
```

# Synchronous Sweep Measurements

This section explains an example subprogram that performs staircase sweep measurement. This example measures MOSFET Id-Vg characteristics. The subprogram uses the synchronous sweep source.

**Table 4-5 Synchronous Sweep Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), 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
 Ag415x.SetSwitch(t(3), 1)             'SMU4: substrate
 Ag415x.SetSwitch(t(2), 1)             'SMU3: source
 Ag415x.SetSwitch(t(1), 1)             'SMU2: gate
 Ag415x.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              '34

 Ag415x.SetIv(t(1), Hp4156b.ModeEnum4.SingleLinearV, 0, vpri1, vpri2, nop1, hold, delay,
s_delay, i1comp, p1comp)
 Ag415x.SetSweepSync(t(0), Hp4156b.ModeEnum.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 34 | Enables measurement channels. And declares variables and sets the value. |
| 36 to 37 | Sets the primary sweep source and the synchronous sweep source. |

```
 Ag415x.ErrorQuery(err, msg)                                                    '38
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 Ag415x.Force(t(3), Hp4156b.ModeEnum.VoltageOutput, 0, vcon1, 0.05, 0)          '41
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, vcon2, 0.05, 0)
 Ag415x.SweepIv(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, 0, rep, sc, md, st)
 If rep <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1                                                          '46
  data(j, i) = Chr(13) & Chr(10) & sc(i) & "," & md(i) * 1000 & "," & st(i)
 Next i

 Ag415x.ZeroOutput(0)                                                           '50
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '53
 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 44 | Applies voltage to device 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. |
| 46 to 48 | Stores the measured data into the *data* valiable. |
| 50 to 51 | 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. |
| 53 to 55 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Vg (V), Id (mA), Status
0,-0.000172985,0
0.3,2.25005,0
0.6,4.73989,0
0.9,7.47392,0
1.2,10.42244,0
1.5,13.5558,0
1.8,16.877,0
2.1,20.3577,0
2.4,23.9656,0
2.699,27.7044,0
3,31.5447,0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Sweep Measurements

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

**Table 4-6 Multi-Channel Sweep Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), Status_b, Ic (mA), Status_c, hfe"
 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 = {2, 3, 1}                                              '13
 Ag415x.SetSwitch(t(2), 1)   'SMU1: emitter
 Ag415x.SetSwitch(t(1), 1)   'SMU3: collector
 Ag415x.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 = {0.0001, 0.001} 'limited auto ranging (0.1 mA, 1 mA)
 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          '35

 Ag415x.SetInteg(Hp4156b.TableEnum.IntegTimeShort, 0.0001, 2)
 Ag415x.SetIv(t(0), Hp4156b.ModeEnum4.SingleLinearV, 0, vb1, vb2, nop1, hold, delay,
s_delay, ibcomp, pcomp)
```

| 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. |
| 37 to 38 | Sets the A/D converter integration time, and sets the primary sweep source. |

```
 Ag415x.ErrorQuery(err, msg)                                                    '39
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, ve, iecomp, 0)           '42
 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, vc, iccomp, 0)
 Ag415x.SweepMiv(mch, mode, range, rep, sc, md, st)
 If rep <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1                                                          '47
  data(j, i) = Chr(13) & Chr(10) & sc(i) & ", " & md(2 * i) * 1000 & ", "
  data(j, i) = data(j, i) & st(2 * i) & ", " & md(2 * i + 1) * 1000 & ", "
  data(j, i) = data(j, i) & st(2 * i + 1) & ", " & md(2 * i + 1) / md(2 * i)
 Next i

 Ag415x.ZeroOutput(0)                                                           '53
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '56
 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 40 | Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 42 to 45 | Applies voltage to device 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 51 | Stores the measured data into the *data* valiable. |
| 53 to 54 | 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. |
| 56 to 58 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Vb (V), Ib (mA), Status_b, Ic (mA), Status_c, hfe
0.2999, -3E-06, 0, -6E-06, 0, 2
0.3499, -3E-06, 0, -3.3E-05, 0, 11
0.4, 2.3E-06, 0, -6E-06, 0, -2.60869565217391
0.45, 5.1E-06, 0, 0.00021, 0, 41.1764705882353
0.5, 1.33E-05, 0, 0.001864, 0, 140.15037593985
0.55, 6.8E-05, 0, 0.012705, 0, 186.838235294118
0.6, 0.0004727, 0, 0.087561, 0, 185.235878993019
0.65, 0.003016, 0, 0.597003, 0, 197.945291777188
0.7, 0.0188088, 0, 3.81314, 0, 202.731700055293
0.75, 0.0953332, 0, 18.5294, 0, 194.364607502948
0.8, 0.318431, 0, 54.3315, 0, 170.622521048516

Data save completed.

Do you want to perform measurement again?
```

# Pulsed Spot Measurements

This section explains an example subprogram that performs pulsed spot measurement. This example measures MOSFET drain current.

**Table 4-7 Pulsed Spot Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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\data6.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
 Ag415x.SetSwitch(t(3), 1)        'SMU4: substrate
 Ag415x.SetSwitch(t(2), 1)        'SMU3: source
 Ag415x.SetSwitch(t(1), 1)        'SMU2: gate
 Ag415x.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

 Ag415x.SetFilter(t(1), Hp4156b.StateEnum2.FilterOff)                        '30
 Ag415x.SetPbias(t(1), Hp4156b.ModeEnum5.VoltagePulse, 2, base, vg, width, period,
hold, igcomp)
 Ag415x.ErrorQuery(err, msg)
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 28 | 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. |

```
 Ag415x.Force(t(3), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)              '35
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)
 Ag415x.Force(t(0), Hp4156b.ModeEnum.VoltageOutput, 2, vd, idcomp, 0)
 Ag415x.MeasureP(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, 0, meas, status)

 data(j, i) = Chr(13) & Chr(10) & meas * 1000 & ", " & status                   '40

 Ag415x.ZeroOutput(0)                                                           '42
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '45
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 35 to 38 | Applies voltage to device and performs pulsed spot measurement. |
| 40 | Stores the measured data into the *data* valiable. |
| 42 to 43 | 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. |
| 45 to 46 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Id (mA), Status
3.8789, 0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Pulsed Spot Measurements

This section explains an example subprogram that performs multi channel pulsed spot measurement. This example measures bipolar transistor collector current and base current.

**Table 4-8 Multi-Channel Pulsed Spot Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), Status_c, Ib (mA), Status_b, hfe"
 Dim fname As String = "C:\Agilent\data\data7.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
 Ag415x.SetSwitch(t(2), 1)        'SMU3: collector
 Ag415x.SetSwitch(t(1), 1)        'SMU2: base
 Ag415x.SetSwitch(t(0), 1)        'SMU1: emitter

 Dim vc As Double = 0
 Dim vb As Double = 0
 Dim iccomp As Double = 0.1
 Dim ibcomp As Double = 0.1
 Dim mch() As Integer = {t(2), t(1), 0}     'collector, base
 Dim mode() As Integer = {1, 1}             'current measurement
 Dim range() As Double = {0, 0}             'auto range
 Dim eod As Integer
 Dim dtype As Integer
 Dim mdata As Double
 Dim stat As Integer
 Dim ch As Integer
 Dim md(2) As Double

 Ag415x.SetFilter(t(0), Hp4156b.StateEnum2.FilterOff)                        '32
 Ag415x.Cmd("PT 0,0.005,0.01,0,1")         'sets pulse timing parameters
 Ag415x.Cmd("PV 1,0,0,-0.8,0.1")           'sets pulse voltage source
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 30 | Enables measurement channels. And declares variables and sets the value. |
| 32 to 34 | Sets the filter off for the pulse output channel and sets the pulse voltage source. |

```
 Ag415x.ErrorQuery(err, msg)                                                    '36
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, vb, ibcomp, 0)           '39
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, vc, iccomp, 0)
 Ag415x.StartMeasure(Hp4156b.MeasTypeEnum.MmPspot, mch, mode, range,
Hp4156b.SourceEnum.WithoutSourceDataOutput)

 Ag415x.ReadData(eod, dtype, mdata, stat, ch)                                   '43
 md(0) = mdata
 data(j, i) = Chr(13) & Chr(10) & mdata * 1000 & ", " & stat
 Ag415x.ReadData(eod, dtype, mdata, stat, ch)
 md(1) = mdata
 data(j, i) = data(j, i) & ", " & mdata * 1000 & ", " & stat
 data(j, i) = data(j, i) & ", " & md(0) / md(1)

 Ag415x.ZeroOutput(0)                                                           '51
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '54
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 36 to 37 | Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 39 to 41 | Applies voltage to device and performs multi channel pulsed spot measurement. |
| 43 to 49 | Reads measurement data, and stores the data into the *data* valiable. |
| 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\data7.txt file (CSV) and displays the data on a message box. |
| 54 to 55 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Ic (mA), Status_c, Ib (mA), Status_b, hfe
42.8595, 0, 0.330382, 0, 129.72710377684

Data save completed.

Do you want to perform measurement again?
```

# Pulsed Sweep Measurements

This section explains an example subprogram that performs pulsed sweep measurement. This example measures bipolar transistor Ic-Vc characteristics.

**Table 4-9 Pulsed Sweep Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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), Status"
 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 = {3, 2, 1}                                              '13
 Ag415x.SetSwitch(t(2), 1)           'SMU1: emitter
 Ag415x.SetSwitch(t(1), 1)           'SMU2: base
 Ag415x.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                  '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
 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

 Ag415x.SetFilter(t(0), Hp4156b.StateEnum2.FilterOff)                        '36
 Ag415x.SetInteg(Hp4156b.TableEnum.IntegTimeShort, 0.0001, 2)
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 34 | Enables measurement channels. And declares variables and sets the value. |
| 36 to 37 | Sets the filter off for the pulse output channel and sets the A/D converter integration time. |

```
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)             '39
 Ag415x.ErrorQuery(err, msg)
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 For j = 0 To nop2 - 1
  Ag415x.SetPiv(t(0), Hp4156b.ModeEnum3.SingleLinearV, 0, base, vc1, vc2, nop1,
hold, width, period, iccomp)                                                   '44
  Ag415x.ErrorQuery(err, msg)
  If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  Ag415x.Force(t(1), Hp4156b.ModeEnum.CurrentOutput, 0, ibo, vbcomp, 0)        '48
  Ag415x.SweepPiv(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, 0, rep, sc, md, st)
  If rep <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                        '52
   data(j, i) = Chr(13) & Chr(10) & ibo * 1000000 & "," & sc(i)
   data(j, i) = data(j, i) & "," & md(i) * 1000 & "," & st(i)
  Next i

  ibo = ibo + d_ib
 Next j

 Ag415x.ZeroOutput(0)                                                          '60
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

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 and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 44 to 46 | Sets the pulsed sweep source and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 48 to 50 | Applies voltage to the device 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. |
| 52 to 55 | Stores the data into the *data* valiable. |
| 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\data8.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), Status
50,0,-0.0503,0
50,0.3,8.5514,0
50,0.6,9.5932,0
50,0.9,9.7166,0
50,1.2,9.7714,0
50,1.5,9.7919,0
50,1.8,9.8262,0
50,2.1,9.8057,0
50,2.4,9.8673,0
50,2.699,9.881,0
50,3,9.8536,0
100,0,-0.0846,0
100,0.3,15.4259,0
100,0.6,18.0373,0
100,0.9,18.6679,0
100,1.2,18.8804,0
100,1.5,18.9351,0
100,1.8,19.0448,0
100,2.1,19.1202,0
100,2.4,19.1339,0
100,2.699,19.1957,0
100,3,19.2094,0
150,0,-0.1394,0
150,0.3,20.4911,0
150,0.6,24.3156,0
150,0.9,26.1593,0
150,1.2,27.0709,0
150,1.5,27.3793,0
150,1.8,27.6741,0
150,2.1,27.6467,0
150,2.4,27.8659,0
150,2.699,27.9139,0
150,3,27.9824,0

Data save completed.

Do you want to perform measurement again?
```

# Multi-Channel Pulsed Sweep Measurements

This section explains an example subprogram that performs multi channel pulsed sweep measurement. This example measures bipolar transistor Ic-Vb and Ib-Vb characteristics.

**Table 4-10 Multi-Channel Pulsed Sweep Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '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 = "Ic (mA), Status_c, Ib (mA), Status_b, Ve (V)"
 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 = {1, 2, 3}                                              '13
 Ag415x.SetSwitch(t(0), 1)           'SMU1: emitter
 Ag415x.SetSwitch(t(1), 1)           'SMU2: base
 Ag415x.SetSwitch(t(2), 1)           'SMU3: collector

 Dim vc As Double = 0
 Dim vb As Double = 0
 Dim iccomp As Double = 0.1
 Dim ibcomp As Double = 0.1
 Dim mch() As Integer = {t(2), t(1), 0}  'measurement channels
 Dim mode() As Integer = {1, 1}          'current measurement mode
 Dim range() As Double = {0, 0}          'auto ranging
 Dim eod As Integer
 Dim dtype As Integer
 Dim mdata As Double
 Dim stat As Integer
 Dim ch As Integer

 Ag415x.SetFilter(t(0), Hp4156b.StateEnum2.FilterOff)                        '31
 Ag415x.SetInteg(Hp4156b.TableEnum.IntegTimeShort, 0.0001, 1)
 Ag415x.Cmd("PT 0,0.005,0.01,0,1")               'sets pulse timing parameters
 Ag415x.Cmd("PWV 1,1,0,0,0,-0.8,11,0.1,0")       'sets pulsed sweep source
```

| 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. |
| 31 to 34 | Sets the filter off for the pulse output channel and sets the A/D converter integration time. Also sets the pulse voltage source. |

```
 Ag415x.ErrorQuery(err, msg)                                                    '36
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err
 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, vb, ibcomp, 0)           '38
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, vc, iccomp, 0)
 Ag415x.StartMeasure(Hp4156b.MeasTypeEnum.MmPsweep, mch, mode, range,
Hp4156b.SourceEnum.SourceDataOutput)

 For i = 0 To nop1 - 1                                                          '42
  Ag415x.ReadData(eod, dtype, mdata, stat, ch)
  data(j, i) = Chr(13) & Chr(10) & mdata * 1000 & "," & stat  't(2)=collector
  Ag415x.ReadData(eod, dtype, mdata, stat, ch)
  data(j, i) = data(j, i) & "," & mdata * 1000 & "," & stat   't(1)=base
  Ag415x.ReadData(eod, dtype, mdata, stat, ch)
  data(j, i) = data(j, i) & "," & mdata                       't(0)=emitter
 Next i

 Ag415x.ZeroOutput(0)                                                           '51
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                      '54
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 36 to 37 | Checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 38 to 40 | Applies voltage to device and performs multi channel pulsed sweep measurement. |
| 42 to 49 | Stores the data into the *data* valiable. |
| 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\data9.txt file (CSV) and displays the data on a message box. |
| 54 to 55 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Ic (mA), Status_c, Ib (mA), Status_b, Ve (V)
-1.21572E-09,0,-3.3938E-10,0,0
7.9188E-10,0,4.3032E-10,0,-0.08
2.9592E-09,0,1.19947E-09,0,-0.16
1.23562E-07,0,9.6629E-09,0,-0.24
1.74463E-06,0,2.70753E-08,0,-0.32
3.76546E-05,0,3.66487E-07,0,-0.4
0.000832422,0,5.73051E-06,0,-0.48
0.0183221,0,0.000107577,0,-0.56
0.400051,0,0.00207699,0,-0.64
7.22489,0,0.0366662,0,-0.72
42.8737,0,0.330252,0,-0.8

Data save completed.

Do you want to perform measurement again?
```

# Staircase Sweep with Pulsed Bias Measurements

This section explains an 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 Ag415x As Hp4156b)                                    '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), 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}                                           '13
 Ag415x.SetSwitch(t(3), 1)           'SMU4: substrate
 Ag415x.SetSwitch(t(2), 1)           'SMU3: source
 Ag415x.SetSwitch(t(1), 1)           'SMU2: gate
 Ag415x.SetSwitch(t(0), 1)           'SMU1: drain

 Dim vd1 As Double = 0
 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                39
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 39 | Enables measurement channels. And declares variables and sets the value. |

```
 Ag415x.SetFilter(t(1), Hp4156b.StateEnum2.FilterOff)                              '41
 Ag415x.Force(t(3), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, 0, 0.05, 0)

 For j = 0 To nop2 - 1
  Ag415x.SetPbias(t(1), Hp4156b.ModeEnum5.VoltagePulse, 0, 0, vg, width, period,
p_hold, igcomp)                                                                   '46
  Ag415x.SetIv(t(0), Hp4156b.ModeEnum4.SingleLinearV, 0, vd1, vd2, nop1, hold,
delay, s_delay, idcomp, p_comp)
  Ag415x.ErrorQuery(err, msg)
  If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  Ag415x.SweepPbias(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, 0, rep, sc, md,
st)                                                                               '51
  If rep <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

  For i = 0 To nop1 - 1                                                           '54
   data(j, i) = Chr(13) & Chr(10) & vg & "," & sc(i) & "," & md(i) * 1000 & "," &
st(i)
  Next i

  vg = vg + d_vg
 Next j

 Ag415x.ZeroOutput(0)                                                             '61
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                        '64
 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 |
|---|---|
| 41 to 43 | Sets the filter off for the pulse output channel and applies voltage to device. |
| 46 to 49 | 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. |
| 51 to 52 | 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. |
| 54 to 56 | Stores the measured data into the *data* valiable. |
| 61 to 62 | 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. |
| 64 to 66 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Vg (V), Vd (V), Id (mA), Status
1,0,-0.0003,0
1,0.3,3.0016,0
1,0.6,5.5649,0
1,0.9,7.6691,0
1,1.2,9.4168,0
1,1.5,10.9109,0
1,1.8,12.316,0
1,2.1,13.3578,0
1,2.4,14.6257,0
1,2.699,15.5852,0
1,3,16.6064,0
2,0,-0.0071,0
2,0.3,3.92,0
2,0.6,7.3332,0
2,0.9,10.4586,0
2,1.2,13.3852,0
2,1.5,15.3453,0
2,1.8,17.4015,0
2,2.1,19.115,0
2,2.4,21.0135,0
2,2.699,21.7194,0
2,3,23.1929,0
3,0,-0.0071,0
3,0.3,4.6945,0
3,0.6,9.122,0
3,0.9,13.1864,0
3,1.2,17.0314,0
3,1.5,19.7592,0
3,1.8,22.6515,0
3,2.1,25.0435,0
3,2.4,26.7638,0
3,2.699,29.615,0
3,3,30.5882,0

Data save completed.

Do you want to perform measurement again?
```

# Sampling Measurements

This section explains an example subprogram that performs sampling measurement. This example measures current of a device that has two high terminals and a low terminal, and calculates the resistance.

**Table 4-12 Sampling Measurement Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 30
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status"
 Dim fname As String = "C:\Agilent\data\data11.txt"
 Dim title As String = "Sampling Measurement Result"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3}                                              '13
 Ag415x.SetSwitch(t(0), 1)           'SMU1: t1
 Ag415x.SetSwitch(t(1), 1)           'SMU2: t2
 Ag415x.SetSwitch(t(2), 1)           'SMU3: low

 Dim base As Double = 0
 Dim bias As Double = 0.1
 Dim icomp As Double = 0.1
 Dim vlout As Double = 0
 Dim ilcomp As Double = 0.1
 Dim hold As Double = 0.1
 Dim interval As Double = 0.05
 Dim mch() As Integer = {t(0), t(1), 0}
 Dim mode() As Integer = {1, 1}
 Dim range() As Double = {0, 0}
 Dim n As Integer = nop1 * 2
 Dim point As Integer
 Dim index(nop1) As Integer
 Dim value(n) As Double
 Dim status(n) As Integer

 Ag415x.SetFilter(Hp4156b.ChannelEnum4.AllSmu, Hp4156b.StateEnum2.FilterOn)  '34
 Ag415x.SetInteg(Hp4156b.TableEnum.IntegTimeShort, 0.0001, 2)
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 32 | Enables measurement channels. And declares variables and sets the value. |
| 34 to 35 | Sets the filter on for the all channels and sets the A/D converter integration time. |

```
 Ag415x.SetSample(hold, interval, nop1)                                    '37
 Ag415x.AddSampleSyncIv(t(0), Hp4156b.ModeEnum.VoltageOutput, 0, base, bias,
icomp)
 Ag415x.AddSampleSyncIv(t(1), Hp4156b.ModeEnum.VoltageOutput, 0, base, bias,
icomp)
 Ag415x.ErrorQuery(err, msg)
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, 0, vlout, ilcomp, 0)   '43
 Ag415x.Sample(mch, mode, range, point, index, value, status)
 Ag415x.ClearSampleSync()
 If point <> nop1 Then Ag415x.ZeroOutput(0) : GoTo Check_err

 For i = 0 To nop1 - 1                                                     '48
  Dim r1 As Double = Math.Round(bias / value(2 * i), 3)
  Dim r2 As Double = Math.Round(bias / value(2 * i + 1), 3)
  data(j, i) = Chr(13) & Chr(10) & index(i) & "," & value(2 * i) * 1000
  data(j, i) = data(j, i) & "," & r1 & "," & value(2 * i + 1) * 1000
  data(j, i) = data(j, i) & "," & r2 & "," & status(i)
 Next i

 Ag415x.ZeroOutput(0)                                                      '56
 save_data(fname, title, val, data, nop1, nop2, Ag415x)

Check_err:                                                                 '59
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
 If point <> nop1 Then MsgBox("No. of data: " & point & " (not " & nop1 & ")",
vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 37 to 41 | Sets the timing parameters, sets the dc voltage sources for sampling measurement, and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 43 to 46 | Applies voltage to device and performs sampling measurement. After the measurement, clears the setup of the dc voltage sources for the sampling measurement. And then, checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. |
| 48 to 54 | Stores the measured data into the *data* valiable. |
| 56 to 57 | 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. |
| 59 to 61 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status
1,9.64451,10.369,9.15948,10.918,0
2,9.64641,10.367,9.16111,10.916,0
3,9.64641,10.367,9.16003,10.917,0
4,9.6456,10.367,9.16112,10.916,0
5,9.64587,10.367,9.1603,10.917,0
6,9.64668,10.366,9.16057,10.916,0
7,9.64669,10.366,9.15976,10.917,0
8,9.64533,10.368,9.16057,10.916,0
9,9.64506,10.368,9.15976,10.917,0
10,9.64614,10.367,9.15894,10.918,0
11,9.6456,10.367,9.15976,10.917,0
12,9.6456,10.367,9.16084,10.916,0
13,9.64641,10.367,9.15976,10.917,0
14,9.6456,10.367,9.16112,10.916,0
15,9.64614,10.367,9.15976,10.917,0
16,9.64505,10.368,9.16003,10.917,0
17,9.64587,10.367,9.16003,10.917,0
18,9.64614,10.367,9.1603,10.917,0
19,9.64669,10.366,9.15948,10.918,0
20,9.6456,10.367,9.16003,10.917,0
21,9.64614,10.367,9.15922,10.918,0
22,9.64505,10.368,9.1603,10.917,0
23,9.64614,10.367,9.15895,10.918,0
24,9.64641,10.367,9.1603,10.917,0
25,9.64668,10.366,9.15895,10.918,0
26,9.64533,10.368,9.15976,10.917,0
27,9.6456,10.367,9.15948,10.918,0
28,9.64587,10.367,9.16084,10.916,0
29,9.64587,10.367,9.16003,10.917,0
30,9.64533,10.368,9.16003,10.917,0

Data save completed.

Do you want to perform measurement again?
```

# Stress Force

This section explains an example subprogram that performs stress measurement. This example measures current of a device that has two high terminals and a low terminal.

**Table 4-13 Stress Force Example**

```
Sub perform_meas(ByVal Ag415x As Hp4156b)                                    '1
 Dim i As Integer = 0
 Dim j As Integer = 0
 Dim nop1 As Integer = 2
 Dim nop2 As Integer = 1
 Dim data(nop2, nop1) As String
 Dim val As String = "Condition, I1 (mA), Status1, I2 (mA), Status2"
 Dim fname As String = "C:\Agilent\data\data12.txt"
 Dim title As String = "Stress Test Result:"
 Dim msg As String = "No error."
 Dim err As Integer = 0

 Dim t() As Integer = {1, 2, 3}                                              '13
 Ag415x.SetSwitch(t(0), 1)           'SMU1: t1
 Ag415x.SetSwitch(t(1), 1)           'SMU2: t2
 Ag415x.SetSwitch(t(2), 1)           'SMU3: low

 Dim range As Double = 0
 Dim base As Double = 0
 Dim stress As Double = 2
 Dim bias As Double = 0.1
 Dim icomp As Double = 0.1
 Dim vlout As Double = 0
 Dim ilcomp As Double = 0.1
 Dim hold As Double = 0
 Dim duration As Double = 5
 Dim period As Double = 0.01
 Dim status As Integer
 Dim md(4) As Double
 Dim st(4) As Integer

 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, range, vlout, ilcomp, 0)  '32
 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, range, bias, icomp, 0)
 Ag415x.Force(t(0), Hp4156b.ModeEnum.VoltageOutput, range, bias, icomp, 0)
 Ag415x.SpotMeas(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, range, md(0), st(0))
 Ag415x.SpotMeas(t(1), Hp4156b.ModeEnum1.CurrentMeasurement, range, md(1), st(1))
 Ag415x.ZeroOutput(0)
```

| Line | Description |
|---|---|
| 2 to 11 | Declares variables used in the program template. And sets the proper values. |
| 13 to 30 | Enables measurement channels. And declares variables and sets the value. |
| 32 to 37 | Applies voltage to device, performs spot measurement, and applies 0 V from the all channels. |

```
 data(j, 0) = Chr(13) & Chr(10) & "Before stress, " & md(0) * 1000 & ", " & st(0)   '38
 data(j, 0) = data(j, 0) & ", " & md(1) * 1000 & ", " & st(1)

 Ag415x.SetStress(hold, Hp4156b.ModeEnum6.DurationMode, duration, period)           '41
 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, range, vlout, ilcomp, 0)
 Ag415x.AddStressSyncIv(1, t(0), 2, range, base, stress, icomp)
 Ag415x.AddStressSyncIv(2, t(1), 2, range, base, stress, icomp)
 Ag415x.ErrorQuery(err, msg)
 If err <> 0 Then Ag415x.ZeroOutput(0) : GoTo Check_err
 Ag415x.Stress(status)
 Ag415x.ZeroOutput(0)
 Ag415x.ClearStressSync()

 Ag415x.Force(t(2), Hp4156b.ModeEnum.VoltageOutput, range, vlout, ilcomp, 0)        '51
 Ag415x.Force(t(1), Hp4156b.ModeEnum.VoltageOutput, range, bias, icomp, 0)
 Ag415x.Force(t(0), Hp4156b.ModeEnum.VoltageOutput, range, bias, icomp, 0)
 Ag415x.SpotMeas(t(0), Hp4156b.ModeEnum1.CurrentMeasurement, range, md(2), st(2))
 Ag415x.SpotMeas(t(1), Hp4156b.ModeEnum1.CurrentMeasurement, range, md(3), st(3))
 Ag415x.ZeroOutput(0)
 data(j, 1) = Chr(13) & Chr(10) & "After stress, " & md(2) * 1000 & ", " & st(2)
 data(j, 1) = data(j, 1) & ", " & md(3) * 1000 & ", " & st(3)

 save_data(fname, title, val, data, nop1, nop2, Ag415x)                            '60

Check_err:                                                                         '62
 If err <> 0 Then MsgBox("Instrument error: " & err & Chr(10) & msg, vbOKOnly, "")
End Sub
```

| Line | Description |
|---|---|
| 38 to 39 | Stores the measured data into the *data* valiable. |
| 41 to 49 | Sets the stress timing parameters, applies voltage to the device, sets the dc voltage sources used for the stress force, and checks if an error occurred. If an error is detected, forces 0 V and goes to Check_err. And then, applies the stress to the device. After the stress, applies 0 V from the all channels and clears the setup of the dc voltage sources for the stress. |
| 51 to 58 | Applies voltage to the device, performs spot measurement, applies 0 V from the all channels, and stores the measured data into the *data* valiable. |
| 60 | 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. |
| 62 to 63 | Displays a message box to show an error message if the error is detected. |

**Measurement Result Example**

```
Condition, I1 (mA), Status1, I2 (mA), Status2
Before stress, 9.80588, 0, 9.27516, 0
After stress, 9.80886, 0, 9.3245, 0

Data save completed.

Do you want to perform measurement again?
```

# 5 Programming Examples for C++ Users

This chapter describes how to create measurement programs using the Agilent 4155/4156 and the 4155/4156 VXI*plug&play* driver, and provides programming examples using Microsoft Visual C++. This chapter contains the following sections:

- “Programming Basics”
- “High-Speed Spot Measurements”
- “Multi-Channel Spot Measurements”
- “Staircase Sweep Measurements”
- “Synchronous Sweep Measurements”
- “Multi-Channel Sweep Measurements”
- “Pulsed Spot Measurements”
- “Multi-Channel Pulsed Spot Measurements”
- “Pulsed Sweep Measurements”
- “Multi-Channel Pulsed Sweep Measurements”
- “Staircase Sweep with Pulsed Bias Measurements”
- “Sampling Measurements”
- “Stress Force”

**NOTE**

## About Program Code

Programming examples are provided as a subprogram that can be run with the project template shown in Table 5-1. The subprograms include the code to perform measurement and to display/save the measurement data. To execute the program, insert the subprogram instead of the perform_meas subprogram in the template.

# Programming Basics

This section provides the basic information for programming using the Agilent 4155/4156 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 4155/4156) 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 hp4156b.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 hp4156b.lib file and the VISA related library files
3. Additional project link library:
   - hp4156b.lib

**Step 4.** Open a source file (.cpp) in the project, and enter a program code as template. See Table 5-1 for example. The example uses 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   "hp4156b.h"

void check_err (ViSession vi, ViStatus ret) {                          /* 6 */
 ViInt32    inst_err;
 ViChar     err_msg[256];

 if(VI_SUCCESS > ret) {
    if ( hp4156b_INSTR_ERROR_DETECTED == ret ) {
       hp4156b_error_query(vi, &inst_err, err_msg);
       printf("Instrument Error: %ld\n %s\n", inst_err, err_msg);
      }
    else {
       hp4156b_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 4155/4156 VXI*plug&play* driver. The header files contain various necessary information such as function declaration and macro definitions.<br><br>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 = hp4156b_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 ) {
       hp4156b_error_message(vi, ret, err_msg);
       printf("Error: %ld\n %s\n", ret, err_msg);
      }
    exit (ret);
   }                                                                        /* 41 */

 ret = hp4156b_reset(vi);                          /* resets 4155/4156       43 */
 ret = hp4156b_timeOut(vi, 60000);                 /* sets 60 second timeout    */
 ret = hp4156b_errorQueryDetect(vi, VI_TRUE);      /* turns on error detection  */

 perform_meas(vi, ret);             /* calls perform_meas subprogram         47 */
 /* ret = hp4156b_cmd(vi, "aa");       sends an invalid command                */
 /* check_err(vi, ret);                checks check_err subprogram operation   */

 /* Closing the session
 ret = hp4156b_close(vi);                                                   52 */
 check_err(vi, ret);

 return VI_SUCCESS;                                                       /* 55 */
}
```

| Line | Description |
|---|---|
| 33 | Establishes the software connection with the Agilent 4155/4156. The above example is for the Agilent 4155/4156 on the GPIB address 17. Confirm the GPIB address of your 4155/4156, and set the address correctly instead of “17”. |
| 34 to 41 | Checks the status returned by the hp4156b_init function. Displays the error message and stops the program execution if an error status is returned. |
| 43 to 45 | Resets the Agilent 4155/4156, 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 4155/4156. |
| 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 4155/4156 VXI*plug&play* driver functions:

- hp4156b_setSwitch to enable/disable the source/measurement channels
- hp4156b_force, hp4156b_setIv, etc. to set source outputs
- hp4156b_spotMeas, hp4156b_sweepIv, etc. to perform measurements
- hp4156b_zeroOutput to disable source outputs

**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 Measurements

This section explains an example subprogram that performs 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 =  3;   /* SMU3 */
bulk =    4;   /* SMU4 */                             /* 11 */

ret = hp4156b_setSwitch(vi, drain, 1);                  /* 13 */
ret = hp4156b_setSwitch(vi, gate, 1);
ret = hp4156b_setSwitch(vi, source, 1);
ret = hp4156b_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 = hp4156b_force(vi, bulk, hp4156b_VF_MODE, 0, 0, 0.1, 0);                /* 31 */
ret = hp4156b_force(vi, source, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_force(vi, gate, hp4156b_VF_MODE, 2, vg, igcomp, 0);
ret = hp4156b_force(vi, drain, hp4156b_VF_MODE, 2, vd, idcomp, 0);
check_err (vi, ret);                                                         /* 35 */

ret = hp4156b_spotMeas(vi,drain,hp4156b_IM_MODE,0,&meas,&status);
check_err (vi, ret);                                                         /* 38 */

ret = hp4156b_zeroOutput(vi, hp4156b_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 = hp4156b_setSwitch(vi, hp4156b_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 = 13.441500 mA (at 1.5 V)
Vg = 1.5 V
```

# Multi-Channel Spot Measurements

This section explains an example subprogram that performs 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 = hp4156b_setSwitch(vi, emitter, 1);               /* 10 */
ret = hp4156b_setSwitch(vi, base, 1);
ret = hp4156b_setSwitch(vi, collector, 1);
check_err (vi, ret);                                   /* 13 */

ViReal64    vc;                                        /* 15 */
ViReal64    vb;
ViReal64    iccomp;
ViReal64    ibcomp;

vc =     3;
iccomp = 0.1;
vb =     0.7;
ibcomp = 0.01;

ViInt32     mch[3];
ViInt32     mode[2];
ViReal64    range[2];
ViReal64    md[2];
ViInt32     st[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;
                                                                          /* 38 */
ret = hp4156b_force(vi, emitter, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_force(vi, base, hp4156b_VF_MODE, 0, vb, ibcomp, 0);
ret = hp4156b_force(vi, collector, hp4156b_VF_MODE, 0, vc, iccomp, 0);
check_err (vi, ret);                                                      /* 42 */

ret = hp4156b_measureM(vi, mch, mode, range, &md[0], &st[0]);
check_err (vi, ret);                                                      /* 45 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                             /* 47 */
check_err (vi, ret);                                                      /* 48 */

printf("Ic = %8.6f mA\n", md[0] * 1000);
printf("Ib = %8.6f mA\n", md[1] * 1000);
printf("hfe = %10.6f \n", md[0]/md[1]);                                   /* 52 */

ret = hp4156b_setSwitch(vi, hp4156b_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. |
| 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
Ib = 0.018970 mA
hfe = 202.767528
```

# Staircase Sweep Measurements

This section explains an example subprogram that performs staircase sweep measurement. The following subprogram performs I-V measurement and save the measurement results (MOSFET Id-Vd characteristics) into a file.

**Table 5-4** **Staircase Sweep Measurement Example**

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{

ViInt32   drain =    1;  /* SMU1 */                     /* 4 */
ViInt32   gate =     2;  /* SMU2 */
ViInt32   source =   3;  /* SMU3 */
ViInt32   bulk =     4;  /* SMU4 */

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  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), Status";
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 = hp4156b_setSwitch(vi, drain, 1);                                    /* 38 */
ret = hp4156b_setSwitch(vi, gate, 1);
ret = hp4156b_setSwitch(vi, source, 1);
ret = hp4156b_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                                      /* 42 */

                                                                          /* 44 */
ret = hp4156b_force(vi, bulk, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_force(vi, source, hp4156b_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                               /* 48 */
   dvg[j] = (j + 1) * vg / nop2;
   ret = hp4156b_force(vi, gate, hp4156b_VF_MODE, 0, dvg[j], igcomp, 0);
   ret = hp4156b_setIv(vi, drain, hp4156b_SWP_VF_SGLLIN, 0, 0, vd, nop1, hold,
delay, s_delay, idcomp, p_comp);
   check_err (vi, ret);

   ret = hp4156b_sweepIv(vi, drain, hp4156b_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[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 = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
      ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                        /* 67 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                             /* 69 */
check_err (vi, ret);
```

| Line | Description |
|---|---|
| 38 to 41 | Enables measurement channels. |
| 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, %d\n", dvg[j], sc[i], md[i] * 1000,
st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                         /* 100 */

ret = hp4156b_setSwitch(vi, hp4156b_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), Status
1.00, 0.00, -0.000114, 0
1.00, 0.30,  3.180000, 0
1.00, 0.60,  5.850000, 0
1.00, 0.90,  8.085500, 0
1.00, 1.20,  9.972000, 0
1.00, 1.50, 11.625000, 0
1.00, 1.80, 13.085000, 0
1.00, 2.10, 14.410000, 0
1.00, 2.40, 15.595000, 0
1.00, 2.70, 16.690000, 0
1.00, 3.00, 17.680000, 0
2.00, 0.00, -0.000117, 0
2.00, 0.30,  4.168000, 0
2.00, 0.60,  7.882000, 0
2.00, 0.90, 11.150500, 0
2.00, 1.20, 13.975000, 0
2.00, 1.50, 16.425000, 0
2.00, 1.80, 18.540000, 0
2.00, 2.10, 20.420000, 0
2.00, 2.40, 22.080000, 0
2.00, 2.70, 23.580000, 0
2.00, 3.00, 24.950000, 0
3.00, 0.00, -0.000114, 0
3.00, 0.30,  5.028500, 0
3.00, 0.60,  9.638000, 0
3.00, 0.90, 13.825000, 0
3.00, 1.20, 17.570000, 0
3.00, 1.50, 20.905000, 0
3.00, 1.80, 23.830000, 0
3.00, 2.10, 26.405000, 0
3.00, 2.40, 28.670000, 0
3.00, 2.70, 30.695000, 0
3.00, 3.00, 32.505000, 0
```

# Synchronous Sweep Measurements

The following subprogram performs I-V measurement and saves the measurement results (MOSFET Id-Vg characteristics) into a file. The subprogram uses the synchronous sweep source set by the hp4156b_setSweepSync function.

**Table 5-5 Synchronous Sweep Measurement Example**

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32   drain =    1;  /* SMU1 */                     /* 3 */
ViInt32   gate =     2;  /* SMU2 */
ViInt32   source =   3;  /* SMU3 */
ViInt32   bulk =     4;  /* SMU4 */
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];
ViInt32   i;
ViChar    f_name[] = "C:\Agilent\data\data2.txt";
ViChar    head1[] = "Vg (V),  Id (mA), Status";
ViChar    msg1[] = "Saving data...";
ViChar    msg2[] = "Data save completed.";
ViChar    c = '\n';                                     /* 26 */
ret = hp4156b_setSwitch(vi, drain, 1);                  /* 27 */
ret = hp4156b_setSwitch(vi, gate, 1);
ret = hp4156b_setSwitch(vi, source, 1);
ret = hp4156b_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                    /* 31 */
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 26 | Declares variables, and defines the value. |
| 27 to 30 | Enables measurement channels. |
| 31 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |

```
ret = hp4156b_force(vi, bulk, hp4156b_VF_MODE, 0, 0, 0.1, 0);              /* 33 */
ret = hp4156b_force(vi, source, hp4156b_VF_MODE, 0, 0, 0.1, 0);

ret = hp4156b_setIv(vi, gate, hp4156b_SWP_VF_SGLLIN, 0, 0, vg, nop, hold, delay,
s_delay, igcomp, pgcomp);
check_err (vi, ret);                                                       /* 37 */

ret = hp4156b_setSweepSync(vi, drain, hp4156b_VF_MODE, 0, 0, vd, idcomp, pdcomp);
check_err (vi, ret);

ret = hp4156b_sweepIv(vi, drain, hp4156b_IM_MODE, 0, &rep, &sc[0], &md[0], &st[0]);
check_err (vi, ret);                                                       /* 43 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                              /* 45 */
ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
check_err (vi, ret);                                                       /* 47 */

if ( rep != nop ) {                                                        /* 49 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }                                                                         /* 52 */

printf(" Vg (V),  Id (mA)\n");                                             /* 54 */
for (i = 0; i < nop; i++){
   printf(" %4.2f,    %9.6f \n", sc[i], md[i] * 1000);
 }                                                                         /* 57 */
```

| Line | Description |
|---|---|
| 33 to 34 | Applies voltage to device. |
| 36 | Sets the primary sweep source. |
| 39 | Sets the synchronous sweep source by using the hp4156b_setSweepSync function. |
| 42 | Performs staircase sweep measurement by using the hp4156b_sweepIv function. |
| 45 | Sets the specified port to the zero output state. |
| 46 | Disables all ports. |
| 37, 40, 43, and 47 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 49 to 52 | Stops the program execution if the number of returned data is not equal to nop. |
| 54 to 57 | Displays the measurement result data. |

```
FILE *stream;                                                          /* 59 */

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, %d\n", sc[i], md[i] * 1000, st[i]);
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                     /* 75 */
}
```

| Line | Description |
|---|---|
| 59 to 75 | Saves the measurement results into a file (C:\Agilent\data\data2.txt, CSV file). |
| 76 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Vg (V),  Id (mA), Status
0.00,  0.004043, 0
0.30,  2.330500, 0
0.60,  4.904000, 0
0.90,  7.723500, 0
1.20, 10.753000, 0
1.50, 13.975000, 0
1.80, 17.385000, 0
2.10, 20.955000, 0
2.40, 24.660000, 0
2.70, 28.500000, 0
3.00, 32.450000, 0
```

# Multi-Channel Sweep Measurements

This section explains an example subprogram that performs 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 5-6** **Multi-Channel Sweep Measurement Example**

```
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;
ViReal64   width =     0.001;
ViReal64   period =    0.01;
ViReal64   p_hold =    0.1;

ViInt32    mch[3];
ViInt32    mode[2];
ViReal64   range[2];
ViInt32    rep;
ViReal64   sc[11];
ViReal64   md[22];
ViInt32    st[22];

mch[0] =   collector;
mch[1] =   base;
mch[2] =   0;
mode[0] =  1;
mode[1] =  1;
range[0] = 0;
range[1] = 0;                                           /* 37 */
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 37 | Declares variables, and defines the value. |

```
ret = hp4156b_setSwitch(vi, emitter, 1);                                    /* 39 */
ret = hp4156b_setSwitch(vi, base, 1);
ret = hp4156b_setSwitch(vi, collector, 1);
check_err (vi, ret);                                                        /* 42 */

ret = hp4156b_force(vi, emitter, hp4156b_VF_MODE, 0, ve, iecomp, 0);        /* 44 */
ret = hp4156b_force(vi, collector, hp4156b_VF_MODE, 0, vc, iccomp, 0);

ret = hp4156b_setIv(vi, base, hp4156b_SWP_VF_SGLLIN, 0, vb1, vb2, nop, hold, delay,
s_delay, ibcomp, pcomp);
check_err (vi, ret);                                                        /* 48 */

ret = hp4156b_sweepMiv(vi, mch, mode, range, &rep, &sc[0], &md[0], &st[0]);
check_err (vi, ret);                                                        /* 51 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                               /* 53 */
ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
check_err (vi, ret);                                                        /* 55 */

if ( rep != nop ) {                                                         /* 57 */
  printf ("%d measurement steps were returned.\nIt must be %d steps.\n", rep, nop);
  exit (ret);
 }
```

| Line | Description |
|---|---|
| 39 to 41 | Enables measurement channels. |
| 44 to 45 | Applies voltage to device. |
| 47 | Sets the staircase sweep source. |
| 50 | Performs measurement by using the hp4156b_sweepMiv function. |
| 53 | Sets the specified port to the zero output state. |
| 54 | Disables all ports. |
| 42, 48, 51, and 55 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 57 to 60 | Stops the program execution if the number of returned data is not equal to the nop value. |

```
ViInt32    i;                                                          /* 67 */
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\data3.txt";                     /* 74 */
ViChar     head1[] =  "Vb (V), Ic (mA), Ib (mA), hfe, 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, %d, %d\n", sc[i], md[2*i] *
1000, md[2*i+1] * 1000, md[2*i]/md[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" );
 }
}                                                                      /* 96 */
```

| Line | Description |
|---|---|
| 67 to 72 | Displays the measurement result data. |
| 74 to 95 | Saves the measurement results into a file (C:\Agilent\data\data3.txt, CSV file). |
| 96 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Vb (V), Ic (mA), Ib (mA), hfe, Status_c, Status_b
0.30,  0.00000083, -0.00000001, -59.41935484, 0, 0
0.35,  0.00000557,  0.00000005, 123.29646018, 0, 0
0.40,  0.00003837,  0.00000032, 119.64452760, 0, 0
0.45,  0.00026580,  0.00000190, 140.15291326, 0, 0
0.50,  0.00185550,  0.00001155, 160.64935065, 0, 0
0.55,  0.01274500,  0.00007378, 172.73158501, 0, 0
0.60,  0.08796500,  0.00047225, 186.26786660, 0, 0
0.65,  0.60135000,  0.00303550, 198.10574864, 0, 0
0.70,  3.84650000,  0.01897000, 202.76752768, 0, 0
0.75, 18.79500000,  0.09735000, 193.06625578, 0, 0
0.80, 55.71000000,  0.33300000, 167.29729730, 0, 0
```

# Pulsed Spot Measurements

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

**Table 5-7 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 =  3;  /* SMU3 */
bulk =    4;  /* SMU4 */

ViReal64  vd;
ViReal64  vg;
ViReal64  idcomp;
ViReal64  igcomp;
ViReal64  base;
ViReal64  width;
ViReal64  period;
ViReal64  hold;
ViReal64  meas;
ViInt32   status;                                    /* 21 */

ret = hp4156b_setSwitch(vi, drain, 1);               /* 23 */
ret = hp4156b_setSwitch(vi, gate, 1);
ret = hp4156b_setSwitch(vi, source, 1);
ret = hp4156b_setSwitch(vi, bulk, 1);
ret = hp4156b_setFilter(vi, gate, hp4156b_FLAG_OFF);
check_err (vi, ret);                                 /* 28 */
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 21 | Declares variables, and defines the value. |
| 23 to 27 | Enables measurement channels, and sets the filter off for the SMU used for the pulse source. |
| 28 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |

```
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 = hp4156b_force(vi, bulk, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_force(vi, source, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_setPbias(vi, gate, hp4156b_VF_MODE, 2, base, vg,
width, period, hold, igcomp);
ret = hp4156b_force(vi, drain, hp4156b_VF_MODE, 2, vd, idcomp, 0);
check_err (vi, ret);                                                   /* 43 */

ret = hp4156b_measureP(vi, drain, hp4156b_IM_MODE, 0, &meas, &status);
check_err (vi, ret);                                                   /* 46 */

ret = hp4156b_zeroOutput(vi, hp4156b_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 = hp4156b_setSwitch(vi, hp4156b_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
```

# Multi-Channel Pulsed Spot Measurements

This section explains an example subprogram that performs multi channel pulsed spot measurement. The following subprogram will apply voltage to a bipolar transistor, measure Ic and Ib, and display the measurement result data.

**Table 5-8** **Multi-Channel Pulsed 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 */
ViReal64    vc;
ViReal64    vb;
ViReal64    iccomp;
ViReal64    ibcomp;
vc =        0;
vb =        0;
iccomp =    0.1;
ibcomp =    0.1;
ViInt32     mch[3];
ViInt32     mode[2];
ViReal64    range[2];
ViReal64    md[2];
ViInt32     st[2];
ViInt32     eod;
ViInt32     type;
ViReal64    mdata;
ViInt32     stat;
ViInt32     ch;
ret = hp4156b_setSwitch(vi, emitter, 1);                /* 27 */
ret = hp4156b_setSwitch(vi, base, 1);
ret = hp4156b_setSwitch(vi, collector, 1);
ret = hp4156b_setFilter(vi, emitter, hp4156b_FLAG_OFF); /* 30 */
check_err (vi, ret);
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 26 | Declares variables, and defines the value. |
| 27 to 30 | Enables measurement channels, and sets the filter off for the SMU used for the pulse source. |
| 31 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |

```
mch[0] =    collector;                                                  /* 32 */
mch[1] =    base;
mch[2] =    0;
mode[0] =   1;
mode[1] =   1;
range[0] =  0;
range[1] =  0;
ret = hp4156b_cmd(vi, "PT 0,0.005,0.01,0,1");                           /* 39 */
ret = hp4156b_cmd(vi, "PV 1,0,0,-0.8,0.1");
ret = hp4156b_force(vi, base, hp4156b_VF_MODE, 0, vb, ibcomp, 0);       /* 41 */
ret = hp4156b_force(vi, collector, hp4156b_VF_MODE, 0, vc, iccomp, 0);
check_err (vi, ret);                                                    /* 43 */
ret = hp4156b_startMeasure(vi, hp4156b_MM_PSPOT, mch, mode, range,
hp4156b_FLAG_OFF);
ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                           /* 45 */
check_err (vi, ret);
ret = hp4156b_readData(vi, &eod, &type, &mdata, &stat, &ch);            /* 47 */
md[0]=mdata;
st[0]=stat;
ret = hp4156b_readData(vi, &eod, &type, &mdata, &stat, &ch);
md[1]=mdata;
st[1]=stat;
check_err (vi, ret);
printf("Ic = %9.6f mA\n", md[0] * 1000);
printf("Ib = %9.6f mA\n", md[1] * 1000);
ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);                         /* 56 */
check_err (vi, ret);
}
```

| Line | Description |
|---|---|
| 32 to 38 | Defines the value for the variables used for the measurement channel. |
| 39 to 40 | Sets the pulse source timing parameters and voltage output parameters. |
| 41 to 42 | Applies voltage to device. |
| 44 | Performs multi channel pulsed spot measurement. |
| 45 | Sets the specified port to the zero output state. |
| 47 to 55 | Reads the measurement data, and displays the result. |
| 56 | Disables all ports. |
| 43, 46, 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. |
| 58 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Ic = 43.357000 mA
Ib =  0.336762 mA
```

# Pulsed Sweep Measurements

This section explains an example subprogram that performs pulsed sweep measurement and saves the measurement results (bipolar transistor Ic-Vc characteristics) into a file.

**Table 5-9 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  dib[3];
ViInt32   i =          0;
ViInt32   j;
ViInt32   n;

ViChar    f_name[] =   "C:\Agilent\data\data4.txt";
ViChar    head1[] =    "Ib (uA), Vc (V), Ic (mA), Status";
ViChar    msg1[] =     "Saving data...";
ViChar    msg2[] =     "Data save completed.";
ViChar    c = '\n';

ret = hp4156b_setSwitch(vi, emitter, 1);              /* 31 */
ret = hp4156b_setSwitch(vi, base, 1);
ret = hp4156b_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 = hp4156b_setFilter(vi, collector, hp4156b_FLAG_OFF);
ret = hp4156b_force(vi, emitter, hp4156b_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                                    /* 39 */
   dib[j] = (j + 1) * ib / nop2;
   ret = hp4156b_force(vi, base, hp4156b_IF_MODE, 0, dib[j], vbcomp, 0);
   ret = hp4156b_setPiv(vi, collector, hp4156b_SWP_VF_SGLLIN, 0, base, 0, vc, nop1,
hold, width, period, iccomp);
   check_err (vi, ret);

   ret = hp4156b_sweepPiv(vi, collector, hp4156b_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[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 = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
      ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                             /* 58 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                                  /* 60 */
check_err (vi, ret);
```

| Line | Description |
|---|---|
| 36 | Sets the filter off for the SMU used for the pulse source. |
| 37 | Applies voltage to device. |
| 39 to 58 | 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. |
| 60 | Sets the specified port to the zero output state. |
| 34, 43, 46, 55, and 61 | 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");                                    /* 63 */

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);
    }
 }                                                                          /* 70 */

FILE *stream;                                                               /* 72 */

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, %d\n", dib[j] * 1E6, sc[i], md[i] *
1000, st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                          /* 91 */

ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);                             /* 93 */
check_err (vi, ret);

}
```

| Line | Description |
|---|---|
| 63 to 70 | Displays the measurement result data. |
| 72 to 91 | Saves the measurement results into a file (C:\Agilent\data\data4.txt, CSV file). |
| 93 | Disables all ports. |
| 94 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 96 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Ib (uA), Vc (V), Ic (mA), Status
 50.0, 0.00, -0.050000, 0
 50.0, 0.30,  9.015000, 0
 50.0, 0.60,  9.760000, 0
 50.0, 0.90,  9.825000, 0
 50.0, 1.20,  9.840000, 0
 50.0, 1.50,  9.875000, 0
 50.0, 1.80,  9.905000, 0
 50.0, 2.10,  9.950000, 0
 50.0, 2.40,  9.935000, 0
 50.0, 2.70,  9.970000, 0
 50.0, 3.00, 10.010000, 0
100.0, 0.00, -0.095000, 0
100.0, 0.30, 15.765000, 0
100.0, 0.60, 18.245000, 0
100.0, 0.90, 18.910000, 0
100.0, 1.20, 19.030000, 0
100.0, 1.50, 19.105000, 0
100.0, 1.80, 19.200000, 0
100.0, 2.10, 19.250000, 0
100.0, 2.40, 19.310000, 0
100.0, 2.70, 19.385000, 0
100.0, 3.00, 19.420000, 0
150.0, 0.00, -0.145000, 0
150.0, 0.30, 21.140000, 0
150.0, 0.60, 24.710000, 0
150.0, 0.90, 26.660000, 0
150.0, 1.20, 27.505000, 0
150.0, 1.50, 27.800000, 0
150.0, 1.80, 27.935000, 0
150.0, 2.10, 28.050000, 0
150.0, 2.40, 28.205000, 0
150.0, 2.70, 28.285000, 0
150.0, 3.00, 28.330000, 0
```

# Multi-Channel Pulsed Sweep Measurements

This section explains an example subprogram that performs multi channel pulsed sweep measurement. The following subprogram performs I-V measurement and saves the measurement results (bipolar transistor Ic-Ve and Ib-Ve characteristics) into a file.

**Table 5-10 Multi-Channel Pulsed Sweep Measurement Example**

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32    emitter =   1;   /* SMU1 */                  /* 3 */
ViInt32    base =      2;   /* SMU2 */
ViInt32    collector = 4;   /* SMU4 */

ViReal64   vb =        0;
ViReal64   vc =        0;
ViReal64   ibcomp =    0.1;
ViReal64   iccomp =    0.1;
ViInt32    nop =       11;

ViInt32    mch[3];
ViInt32    mode[2];
ViReal64   range[2];
ViInt32    rep;
ViReal64   sc[11];
ViReal64   md1[11];
ViReal64   md2[11];

ViInt32    eod;
ViInt32    type;
ViReal64   mdata;
ViInt32    stat;
ViInt32    ch;

mch[0] =   collector;
mch[1] =   base;
mch[2] =   0;
mode[0] =  1;
mode[1] =  1;
range[0] = 0;
range[1] = 0;                                          /* 33 */
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 33 | Declares variables, and defines the value. |

```
ret = hp4156b_setSwitch(vi, emitter, 1);                                    /* 35 */
ret = hp4156b_setSwitch(vi, base, 1);
ret = hp4156b_setSwitch(vi, collector, 1);
ret = hp4156b_setFilter(vi, emitter, hp4156b_FLAG_OFF);
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, hp4156b_INTEG_TIME_MIN, 2);
check_err (vi, ret);                                                        /* 40 */

ret = hp4156b_cmd(vi, "PT 0,0.005,0.01,0,1");                               /* 42 */
ret = hp4156b_cmd(vi, "PWV 1,1,0,0,0,-0.8,11,0.1,0");
ret = hp4156b_force(vi, base, hp4156b_VF_MODE, 0, vb, ibcomp, 0);           /* 44 */
ret = hp4156b_force(vi, collector, hp4156b_VF_MODE, 0, vc, iccomp, 0);

ret = hp4156b_startMeasure(vi, hp4156b_MM_PSWEEP, mch, mode, range,
hp4156b_FLAG_ON);
check_err (vi, ret);                                                        /* 48 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                               /* 50 */
ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
check_err (vi, ret);                                                        /* 52 */

ViInt32    i;                                                               /* 54 */
for (i = 0; i < nop ; i++){
  ret = hp4156b_readData(vi, &eod, &type, &mdata, &stat, &ch);
  md1[i] = mdata;       /* Data measured by mch[0] = collector */
  ret = hp4156b_readData(vi, &eod, &type, &mdata, &stat, &ch);
  md2[i] = mdata;       /* Data measured by mch[1] = base */
  ret = hp4156b_readData(vi, &eod, &type, &mdata, &stat, &ch);
  sc[i] = mdata;        /* Pulsed sweep source output data */
}                                                                           /* 62 */
```

| Line | Description |
|---|---|
| 35 to 37 | Enables measurement channels. |
| 38 | Sets the filter off for the SMU used for the pulse source. |
| 39 | Sets the A/D converter integration time. |
| 42 to 43 | Sets the pulsed sweep source timing parameters and voltage output parameters. |
| 44 to 45 | Applies voltage to device. |
| 47 | Performs multi channel pulsed sweep measurement. |
| 50 | Sets the specified port to the zero output state. |
| 51 | Disables all ports. |
| 40, 48, and 52 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 54 to 62 | Reads measurement data. |

```
printf(" Ve (V),  Ic (mA),  Ib (mA)\n");
for (i = 0; i < nop; i++){
  printf(" %5.2f,  %11.8f,  %11.8f\n", sc[i], md1[i] * 1000, md2[i] * 1000);
 }

ViChar     f_name[] = "C:\Agilent\data\data5.txt";                    /* 74 */
ViChar     head1[] =  "Ve (V), Ic (mA), Ib (mA)";
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, "%5.2f, %11.8f, %11.8f\n", sc[i], md1[i] * 1000, md2[i] *
1000);
   }
  printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
  printf( "Data file was not closed\n" );
 }
}                                                                     /* 96 */
```

| Line | Description |
|---|---|
| 67 to 72 | Displays the measurement result data. |
| 74 to 95 | Saves the measurement results into a file (C:\Agilent\data\data5.txt, CSV file). |
| 96 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Ve (V), Ic (mA), Ib (mA)
 0.00,  0.00000000,  0.00000000
-0.08,  0.00000000,  0.00000000
-0.16,  0.00000000,  0.00000000
-0.24,  0.00000008,  0.00000000
-0.32,  0.00000169,  0.00000003
-0.40,  0.00003758,  0.00000035
-0.48,  0.00083207,  0.00000569
-0.56,  0.01834160,  0.00010753
-0.64,  0.40003000,  0.00207454
-0.72,  7.24961000,  0.03680280
-0.80, 43.35560000,  0.33661400
```

# Staircase Sweep with Pulsed Bias Measurements

This section explains an example subprogram that performs 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 =   3;  /* SMU3 */
ViInt32   bulk =     4;  /* SMU4 */
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  dvg[3];

ViChar    f_name[] = "C:\Agilent\data\data6.txt";
ViChar    head1[] =  "Vg (V), Vd (V), Id (mA), Status";
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 = hp4156b_setSwitch(vi, drain, 1);                                    /* 38 */
ret = hp4156b_setSwitch(vi, gate, 1);
ret = hp4156b_setSwitch(vi, source, 1);
ret = hp4156b_setSwitch(vi, bulk, 1);
check_err (vi, ret);                                                      /* 42 */

ret = hp4156b_setFilter(vi, gate, hp4156b_FLAG_OFF);                      /* 44 */
ret = hp4156b_force(vi, bulk, hp4156b_VF_MODE, 0, 0, 0.1, 0);
ret = hp4156b_force(vi, source, hp4156b_VF_MODE, 0, 0, 0.1, 0);

for (j = 0; j < nop2; j++){                                               /* 48 */
   dvg[j] = (j + 1) * vg / nop2;
   ret = hp4156b_setPbias(vi, gate, hp4156b_VF_MODE, 0, 0, dvg[j], width, period,
p_hold, igcomp);
   ret = hp4156b_setIv(vi, drain, hp4156b_SWP_VF_SGLLIN, 0, 0, vd, nop1, hold,
delay, s_delay, idcomp, p_comp);
   check_err (vi, ret);

   ret = hp4156b_sweepPbias(vi, drain, hp4156b_IM_MODE, 0, &rep, &sc[i], &md[i],
&st[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 = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
      ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
      check_err (vi, ret);
      exit (ret);
    }
 }                                                                        /* 67 */

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);                             /* 69 */
check_err (vi, ret);
```

| Line | Description |
|---|---|
| 38 to 41 | Enables measurement channels. |
| 44 | Sets the filter off for the SMU used for the pulse source. |
| 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, %d\n", dvg[j], sc[i], md[i] * 1000,
st[i]);
       }
    }
   printf( "%s%c", msg2, c );
 }

if( fclose( stream ) ){
   printf( "Data file was not closed\n" );
 }                                                                         /* 100 */

ret = hp4156b_setSwitch(vi, hp4156b_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\data6.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), Status
1.00, 0.00,  0.005000, 0
1.00, 0.30,  3.170000, 0
1.00, 0.60,  5.835000, 0
1.00, 0.90,  8.040000, 0
1.00, 1.20,  9.905000, 0
1.00, 1.50, 11.530000, 0
1.00, 1.80, 12.965000, 0
1.00, 2.10, 14.270000, 0
1.00, 2.40, 15.425000, 0
1.00, 2.70, 16.495000, 0
1.00, 3.00, 17.460000, 0
2.00, 0.00,  0.005000, 0
2.00, 0.30,  4.165000, 0
2.00, 0.60,  7.875000, 0
2.00, 0.90, 11.135000, 0
2.00, 1.20, 13.945000, 0
2.00, 1.50, 16.370000, 0
2.00, 1.80, 18.470000, 0
2.00, 2.10, 20.320000, 0
2.00, 2.40, 21.950000, 0
2.00, 2.70, 23.430000, 0
2.00, 3.00, 24.780000, 0
3.00, 0.00,  0.000000, 0
3.00, 0.30,  5.035000, 0
3.00, 0.60,  9.650000, 0
3.00, 0.90, 13.835000, 0
3.00, 1.20, 17.575000, 0
3.00, 1.50, 20.895000, 0
3.00, 1.80, 23.810000, 0
3.00, 2.10, 26.355000, 0
3.00, 2.40, 28.615000, 0
3.00, 2.70, 30.615000, 0
3.00, 3.00, 32.410000, 0
```

# Sampling Measurements

This section explains an example subprogram that measures current of a device with two high terminals and a low terminal, and saves the measurement results data.

**Table 5-12** **Sampling Measurement Example**

```
void perform_meas (ViSession vi, ViStatus ret)          /* 1 */
{
ViInt32    t1 =       1;  /* SMU1 */
ViInt32    t2 =       2;  /* SMU2 */
ViInt32    low =      3;  /* SMU3 */
ViReal64   base =     0;
ViReal64   bias =     0.1;
ViReal64   icomp =    0.1;
ViReal64   vlout =    0;
ViReal64   ilcomp =   0.1;
ViReal64   hold =     0.1;
ViReal64   interval = 0.05;
ViInt32    nop =      30;
ViInt32    mch[3];
ViInt32    mode[2];
ViReal64   range[2];
ViInt32    point;
ViInt32    index[30];
ViReal64   value[60];
ViInt32    status[60];
mch[0] =   t1;
mch[1] =   t2;
mch[2] =   0;
mode[0] =  1;
mode[1] =  1;
range[0] = 0;
range[1] = 0;                                           /* 27 */
ret = hp4156b_setSwitch(vi, t1, 1);
ret = hp4156b_setSwitch(vi, t2, 1);
ret = hp4156b_setSwitch(vi, low, 1);
ret = hp4156b_setFilter(vi, hp4156b_CH_ALL, hp4156b_FLAG_ON);
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT, 0.0001, 2);
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 27 | Declares variables, and defines the value. |
| 28 to 30 | Enables measurement channels. |
| 31 | Sets the filter on for the all SMUs. |
| 32 | Sets the A/D converter integration time. |

```
ret = hp4156b_setSample(vi, hold, interval, nop);                          /* 34 */
ret = hp4156b_addSampleSyncIv(vi, t1, hp4156b_VF_MODE, 0, base, bias, icomp);
ret = hp4156b_addSampleSyncIv(vi, t2, hp4156b_VF_MODE, 0, base, bias, icomp);
ret = hp4156b_force(vi, low, hp4156b_VF_MODE, 0, vlout, ilcomp, 0);
check_err (vi, ret);

ret = hp4156b_sample(vi, mch, mode, range, &point, &index[0], &value[0],
&status[0]);                                                               /* 40 */
check_err (vi, ret);

ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
check_err (vi, ret);

if ( point != nop ) {                                                      /* 46 */
    printf ("%d measurement data were returned.\nIt must be %d data.\n", point,
nop);
    ret = hp4156b_clearSampleSync(vi);
    ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
    check_err (vi, ret);
    exit (ret);
}

ViInt32  i;                                                                /* 54 */
ViChar   f_name[] = "C:\Agilent\data\data7.txt";
ViChar   head1[] =  "Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status";
ViChar   msg1[] =   "Saving data...";
ViChar   msg2[] =   "Data save completed.";
ViChar   c =        '\n';
```

| Line | Description |
|---|---|
| 34 | Sets the sampling measurement timing parameters and the number of samples. |
| 35 to 36 | Sets the constant voltage sources for the sampling measurement. |
| 37 | Applies voltage to the low terminal of the device. |
| 40 | Performs the sampling measurement using the channels specified by mch. |
| 43 | Sets the specified port to the zero output state. |
| 46 to 52 | Clears the sampling source setup information, disables all ports, and stops the program execution if the number of returned data is not equal to the nop value. |
| 38, 41, 44, and 50 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 54 to 59 | Declares variables, and defines the value. |

```
printf(" Index,  R1 (ohm),  R2 (ohm)\n");                                    /* 61 */
for (i = 0; i < nop; i++){
    printf("   %2d,   %6.3f,    %6.3f \n", index[i], bias/value[2 * i], bias/value[2
* i + 1]);
}                                                                            /* 64 */

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,"%2d,%7.3f,%6.3f,%7.3f,%6.3f, %d\n",index[i],value[2 * i] *
1000, bias/value[2 * i], value[2 * i + 1] * 1000, bias/value[2 * i + 1], status[i]);
}
printf( "%s%c", msg2, c );
}

if( fclose( stream ) ){
    printf( "Data file was not closed\n" );
}                                                                            /* 82 */

ret = hp4156b_clearSampleSync(vi);
ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
check_err (vi, ret);
}
```

| Line | Description |
|---|---|
| 61 to 64 | Displays the measurement result data. |
| 66 to 82 | Saves the measurement results into a file (C:\Agilent\data\data7.txt, CSV file). |
| 84 | Clears the source setup information for the sampling measurement. |
| 85 | Disables all ports. |
| 86 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 87 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Index, I1 (mA), R1 (ohm), I2 (mA), R2 (ohm), Status
 1, 11.136, 8.980, 10.755, 9.298, 0
 2, 11.136, 8.980, 10.759, 9.295, 0
 3, 11.147, 8.971, 10.766, 9.289, 0
 4, 11.136, 8.980, 10.762, 9.292, 0
 5, 11.136, 8.980, 10.759, 9.295, 0
 6, 11.147, 8.971, 10.762, 9.292, 0
 7, 11.143, 8.974, 10.755, 9.298, 0
 8, 11.143, 8.974, 10.769, 9.286, 0
 9, 11.136, 8.980, 10.752, 9.301, 0
10, 11.133, 8.982, 10.759, 9.295, 0
11, 11.147, 8.971, 10.752, 9.301, 0
12, 11.136, 8.980, 10.759, 9.295, 0
13, 11.136, 8.980, 10.755, 9.298, 0
14, 11.147, 8.971, 10.766, 9.289, 0
15, 11.140, 8.977, 10.755, 9.298, 0
16, 11.143, 8.974, 10.762, 9.292, 0
17, 11.153, 8.966, 10.762, 9.292, 0
18, 11.147, 8.971, 10.759, 9.295, 0
19, 11.143, 8.974, 10.755, 9.298, 0
20, 11.143, 8.974, 10.755, 9.298, 0
21, 11.140, 8.977, 10.762, 9.292, 0
22, 11.140, 8.977, 10.755, 9.298, 0
23, 11.140, 8.977, 10.759, 9.295, 0
24, 11.136, 8.980, 10.759, 9.295, 0
25, 11.133, 8.982, 10.749, 9.304, 0
26, 11.136, 8.980, 10.762, 9.292, 0
27, 11.136, 8.980, 10.755, 9.298, 0
28, 11.136, 8.980, 10.766, 9.289, 0
29, 11.133, 8.982, 10.755, 9.298, 0
30, 11.147, 8.971, 10.759, 9.295, 0
```

# Stress Force

This section explains an example subprogram that performs current measurement of a device with two high terminals and a low terminal, applies stress, performs current measurement again, and displays the measurement result data.

**Table 5-13 Stress Force Example**

```
void perform_meas (ViSession vi, ViStatus ret)           /* 1 */
{
ViInt32    t1 =       1;  /* SMU1 */
ViInt32    t2 =       2;  /* SMU2 */
ViInt32    low =      3;  /* SMU3 */
ViReal64   range =    0.0;
ViReal64   base =     0.0;
ViReal64   stress =   2.0;
ViReal64   bias =     0.1;
ViReal64   icomp =    0.1;
ViReal64   vlout =    0;
ViReal64   ilcomp =   0.1;
ViReal64   hold =     0.0;
ViReal64   duration = 5.0;
ViReal64   period =   0.01;
ViInt32    status;
ViReal64   md[2];
ViInt32    st[2];                                        /* 18 */

ret = hp4156b_setSwitch(vi, t1, 1);
ret = hp4156b_setSwitch(vi, t2, 1);
ret = hp4156b_setSwitch(vi, low, 1);
ret = hp4156b_setFilter(vi, hp4156b_CH_ALL, hp4156b_FLAG_ON);
ret = hp4156b_setInteg(vi, hp4156b_INTEG_TBL_SHORT,
hp4156b_INTEG_TIME_MIN, 2);
check_err (vi, ret);
```

| Line | Description |
|---|---|
| 1 | Beginning of the perform_meas subprogram. |
| 3 to 18 | Declares variables, and defines the value. |
| 20 to 22 | Enables measurement channels. |
| 23 | Sets the filter on for the all SMUs. |
| 24 | Sets the A/D converter integration time. |
| 25 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |

```
ret = hp4156b_force(vi, low, hp4156b_VF_MODE, range, vlout, ilcomp, 0);   /* 27 */
ret = hp4156b_force(vi, t1, hp4156b_VF_MODE, range, bias, icomp, 0);
ret = hp4156b_force(vi, t2, hp4156b_VF_MODE, range, bias, icomp, 0);
ret = hp4156b_spotMeas(vi,t1, hp4156b_IM_MODE, range, &md[0], &st[0]);
ret = hp4156b_spotMeas(vi,t2, hp4156b_IM_MODE, range, &md[1], &st[1]);
ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
check_err (vi, ret);

printf("Data before stress:\n");                                          /* 35 */
printf("T1 current = %8.5f [mA], Status = %d \n", md[0] * 1000, st[0]);
printf("T2 current = %8.5f [mA], Status = %d \n", md[1] * 1000, st[1]);
printf("\n");

ret = hp4156b_setStress(vi, hold, hp4156b_STT_TIME, duration, period);   /* 40 */
check_err (vi, ret);
ret = hp4156b_force(vi, low, hp4156b_VF_MODE, range, vlout, ilcomp, 0);
ret = hp4156b_addStressSyncIv(vi, 1, t1, 2, range, base, stress, icomp);
check_err (vi, ret);
ret = hp4156b_addStressSyncIv(vi, 2, t2, 2, range, base, stress, icomp);
check_err (vi, ret);

printf("Stress force in progress: %3.1f [V], %3.1f [sec]\n", stress, duration);
printf("\n");

ret = hp4156b_stress(vi, &status);                                       /* 51 */
check_err (vi, ret);

ret = hp4156b_clearStressSync(vi);
```

| Line | Description |
|---|---|
| 27 to 32 | Applies voltage to device, performs spot measurement, and sets the specified port to the zero output state. |
| 35 to 38 | Displays the measurement result data. |
| 40 | Sets the stress source timing parameters. |
| 42 | Applies voltage to the low terminal of the device. |
| 43 and 45 | Sets the stress source outputs. |
| 51 | Applies stress to the device. |
| 54 | Clears the stress source setup information. |
| 33, 41, 44, 46, and 52 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |

```
ret = hp4156b_force(vi, t1, hp4156b_VF_MODE, range, bias, icomp, 0);      /* 56 */
ret = hp4156b_force(vi, t2, hp4156b_VF_MODE, range, bias, icomp, 0);
ret = hp4156b_spotMeas(vi,t1, hp4156b_IM_MODE, range, &md[0], &st[0]);
ret = hp4156b_spotMeas(vi,t2, hp4156b_IM_MODE, range, &md[1], &st[1]);
ret = hp4156b_zeroOutput(vi, hp4156b_CH_ALL);
check_err (vi, ret);

printf("Data after stress:\n");                                           /* 63 */
printf("T1 current = %8.5f [mA], Status = %d \n", md[0] * 1000, st[0]);
printf("T2 current = %8.5f [mA], Status = %d \n", md[1] * 1000, st[1]);

ret = hp4156b_setSwitch(vi, hp4156b_CH_ALL, 0);
check_err (vi, ret);
}
```

| Line | Description |
|---|---|
| 56 to 60 | Applies voltage to device, performs spot measurement, and sets the specified port to the zero output state. |
| 63 to 65 | Displays the measurement result data. |
| 67 | Disables all ports. |
| 61 and 68 | Calls the check_err subprogram (shown in Table 5-1) to check if an error status is returned for the previous line. |
| 69 | End of the perform_meas subprogram. |

**Measurement Result Example**

```
Data before stress:
T1 current = 11.09860 [mA], Status = 0
T2 current = 10.75880 [mA], Status = 0

Stress force in progress: 2.0 [V], 5.0 [sec]

Data after stress:
T1 current = 11.10200 [mA], Status = 0
T2 current = 10.76220 [mA], Status = 0
```

# 6 Programming Examples for VEE Users

This chapter describes how to create measurement programs using the 4155/4156 VXI*plug&play* driver and Agilent VEE, and provides programming examples. This chapter contains the following sections:

- “Programming Basics”
- “High-Speed Spot Measurements”
- “Multi-Channel Spot Measurements”
- “Staircase Sweep Measurements”
- “Synchronous Sweep Measurements”
- “Multi-Channel Sweep Measurements”
- “Pulsed Spot Measurements”
- “Multi-Channel Pulsed Spot Measurements”
- “Pulsed Sweep Measurements”
- “Multi-Channel Pulsed Sweep Measurements”
- “Staircase Sweep with Pulsed Bias Measurements”
- “Sampling Measurements”
- “Stress Force”

# Programming Basics

This section covers the following topics.

- “Registrating the Driver on Agilent VEE”
- “Basic Objects to Control the Instrument”
  - “To display the To/From object”
  - “To define transactions in the To/From object”
  - “To set input parameters”
  - “To use the Help function”
  - “To use input variables”
  - “To create output terminals in the To/From object”
  - “To display/connect the Data object”
  - “To display/connect the Display object”
- “Debugging Your Program”
- “Restrictions When Using the Driver with Agilent VEE”

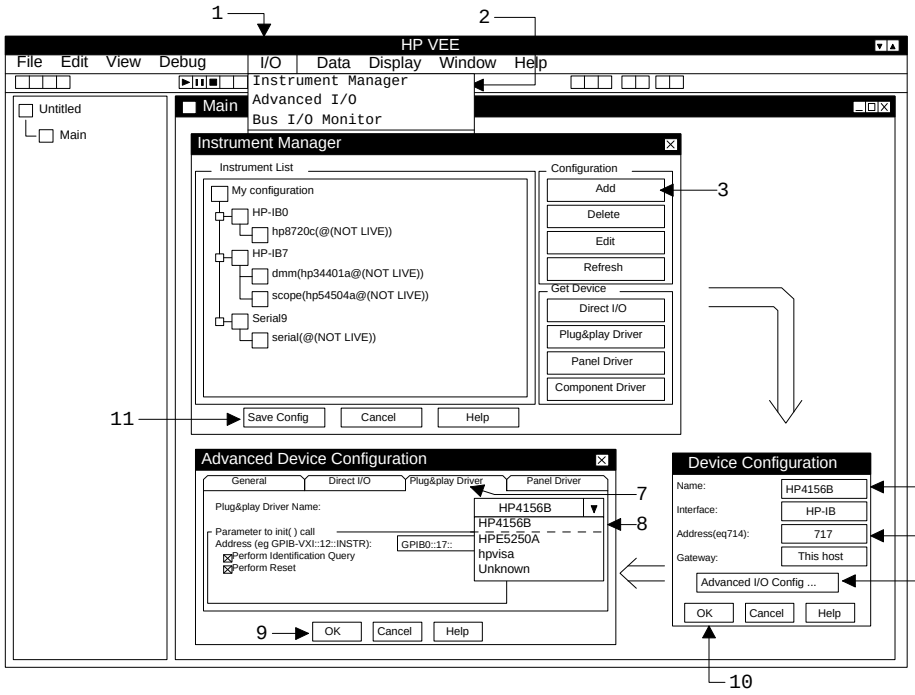
## Registrating the Driver on Agilent VEE

To use the VXI*plug&play* driver on Agilent VEE, register the driver as described below and as shown in Figure 6-1 on page 6-5.

1. Click the I/O menu.
2. Select Instrument Manager from the I/O menu. The Instrument Manager dialog box is displayed. The dialog box lists the available devices (instruments). If this is the first time using Agilent VEE, only off-line (NOT LIVE) devices are shown in the Instrument List.
3. Click Add. The Device Configuration dialog box is displayed.
4. Enter the device name in the Name field. The example shown in Figure 6-1 sets "HP4156B".
5. Enter the GPIB address for the device in the Address field. The example shown in Figure 6-1 sets "717".
6. Click Advanced I/O Config. The Advanced Device Configuration dialog box is displayed.
7. Click the Plug&play Driver tab.
8. Select HP4156B in the Plug&play Driver Name field to configure the 4155/4156 driver. If the driver is not installed properly, "HP4156B" is not available in this field. Install the driver properly at this time.
9. Click OK to close the Advanced Device Configuration dialog box.
10. Click OK to close the Device Configuration dialog box.
11. Click Save Config to save the configuration of the drivers. The Instrument Manager dialog box is closed.

You can now use the VXI*plug&play* driver for the 4155/4156.

**Figure 6-1 Registering the Driver on Agilent VEE**

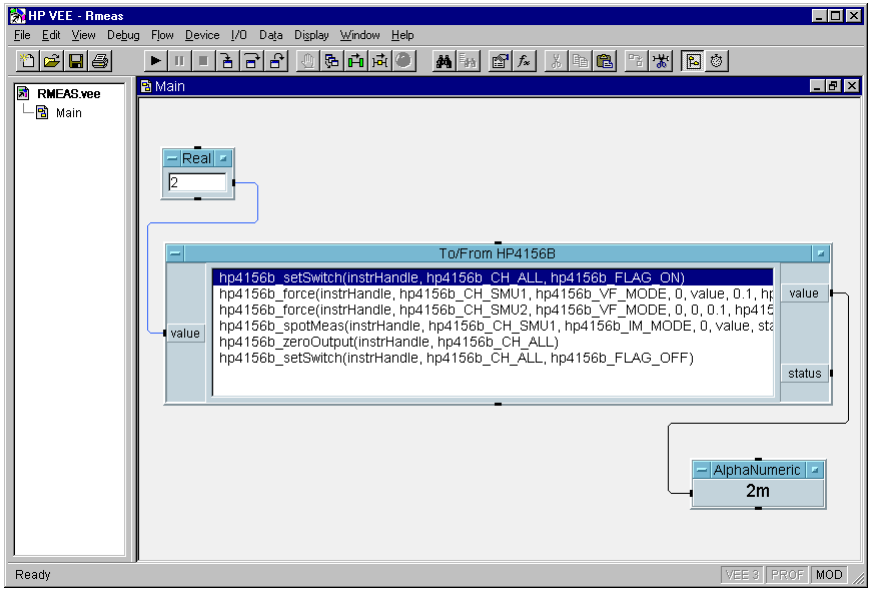


# Basic Objects to Control the Instrument

You can create programs using Agilent VEE and the VXI*plug&play* driver, as shown in the following steps. In this procedure you use only three objects; To/From, Data, and Display, shown in Figure 6-2.

1. Display the To/From object for the VXI*plug&play* driver.
2. Define the transactions (functions of the driver) in the To/From object.
3. Set the input parameters for the transaction.
4. (Optional: Use a variable for the input parameter.)
5. Repeat steps 2, 3, and 4 to complete the To/From object.
6. Connect the input terminals of the To/From object to the Data object.
7. Connect the output terminals of the To/From object to the Display object.
8. Complete the Agilent VEE program.

**Figure 6-2** **Basic Objects of Agilent VEE**



The To/From HP4156B object, in Figure 6-2, defines the following transactions (functions of the *plug&play* driver) to measure the current flow to a resistor.

| | |
|---|---|
| hp4156b_setSwitch | This function controls the 4155/4156 output switch. |
| hp4156b_force | This function forces dc voltage or current. |
| hp4156b_spotMeas | This function executes a spot measurement. |
| hp4156b_zeroOutput | This function disables the 4155/4156 output. |

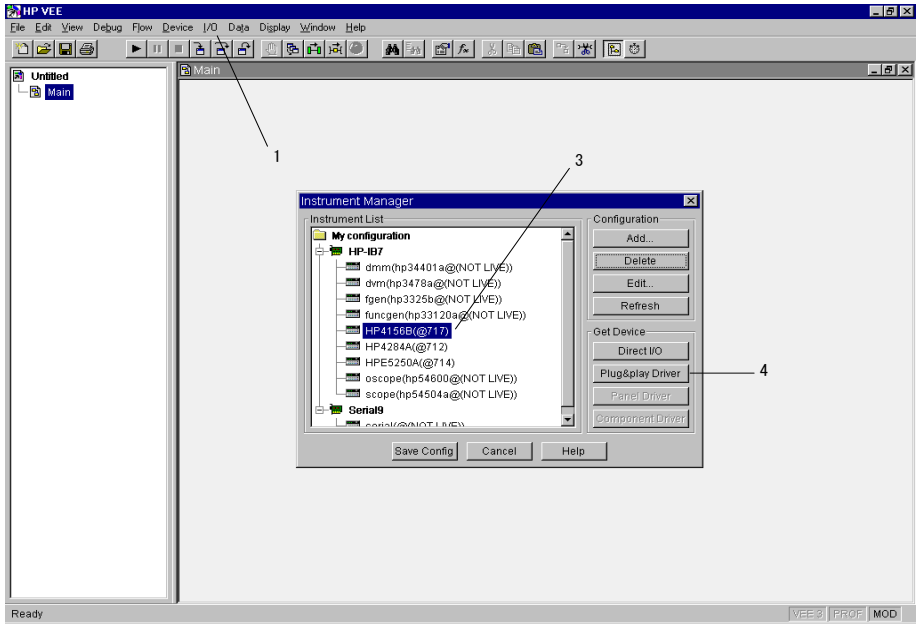
## To display the To/From object

You can display the To/From object as shown below.

1. Click I/O menu.
2. Select Instrument Manager to display the Instrument Manager dialog box.
3. Select HP4156B in the Instrument List.
4. Click Plug&play Driver.

The Instrument Manager dialog box is then closed, and the To/From HP4156B object will be displayed by moving the mouse pointer to the appropriate point, then clicking the left mouse button.

**Figure 6-3** **To Display the To/From Object**

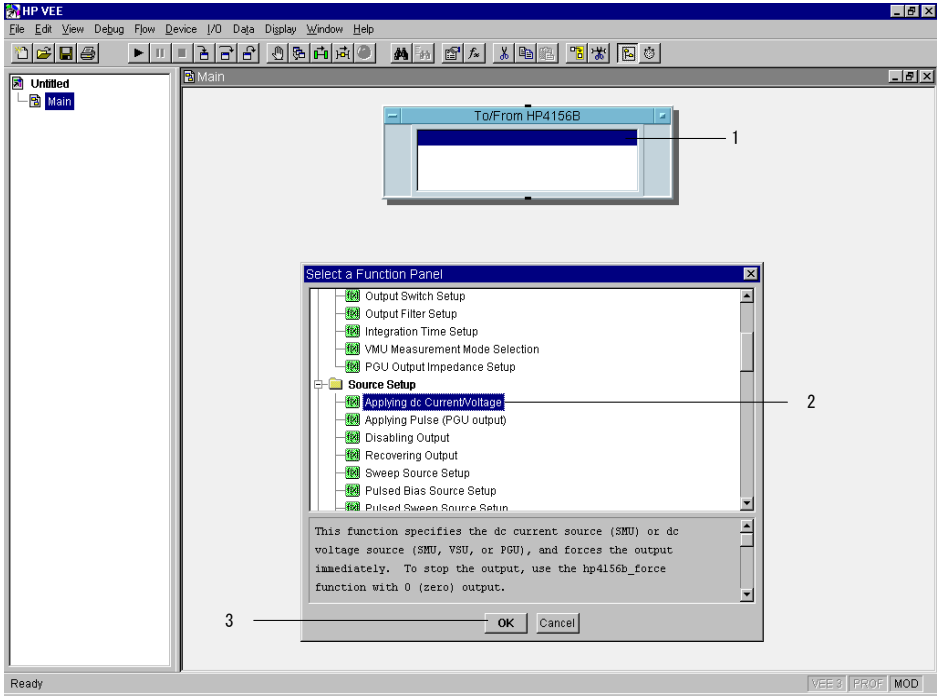


## To define transactions in the To/From object

You can define transactions (functions of *plug&play* driver) as shown in the following example.

1. Double click the blue stripe on the To/From object. The Select a Function Panel dialog box is displayed. The dialog box lists the functions available for the instrument, and displays the Help message for the selected function.
2. Select the function you want to add to the To/From object. Figure 6-4 selects the “Applying dc Current/Voltage” function, and displays the Help message for that function.
3. Click OK. The Select a Function Panel dialog box is closed, and the Edit Function Panel dialog box is displayed. See Figure 6-5 on page 6-9.

**Figure 6-4** **To Define Transactions**




**NOTE**

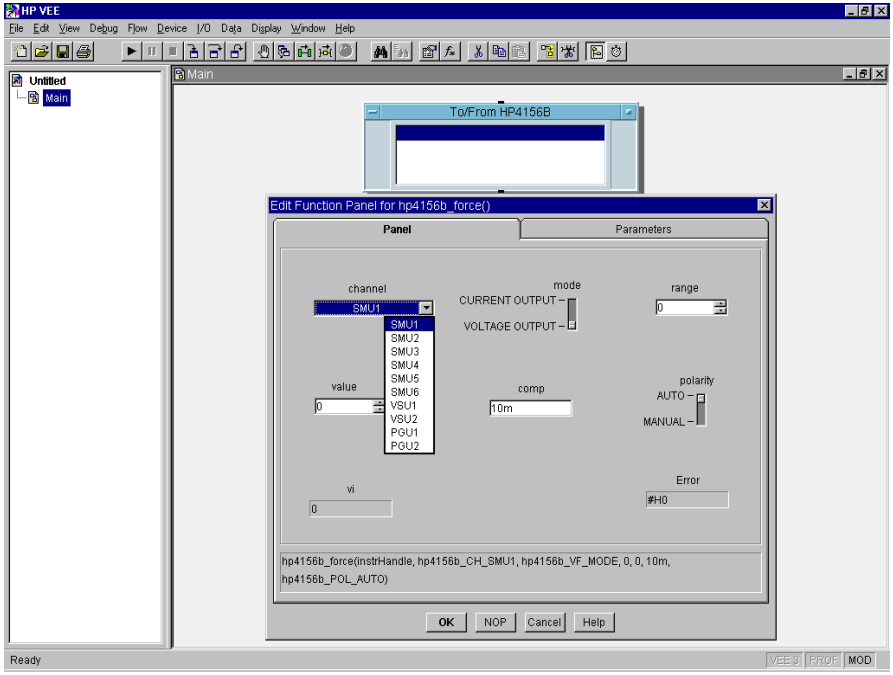
To add, insert, cut, copy, or paste the transaction, click the right mouse button on the To/From object, then select Add Trans, Insert Trans, Cut Trans, Copy Trans, or Paste Trans.

## To set input parameters

You can set the input parameter value using the Edit Function Panel dialog box. Figure 6-5 sets the following values for the input parameters of the hp4156b_force function, which forces dc current or voltage.

| | |
|---|---|
| channel | SMU1 |
| mode | VOLTAGE OUTPUT |
| range (output range) | 0 (auto range) |
| value | 0 V |
| compliance | 10 mA |
| polarity | AUTO |

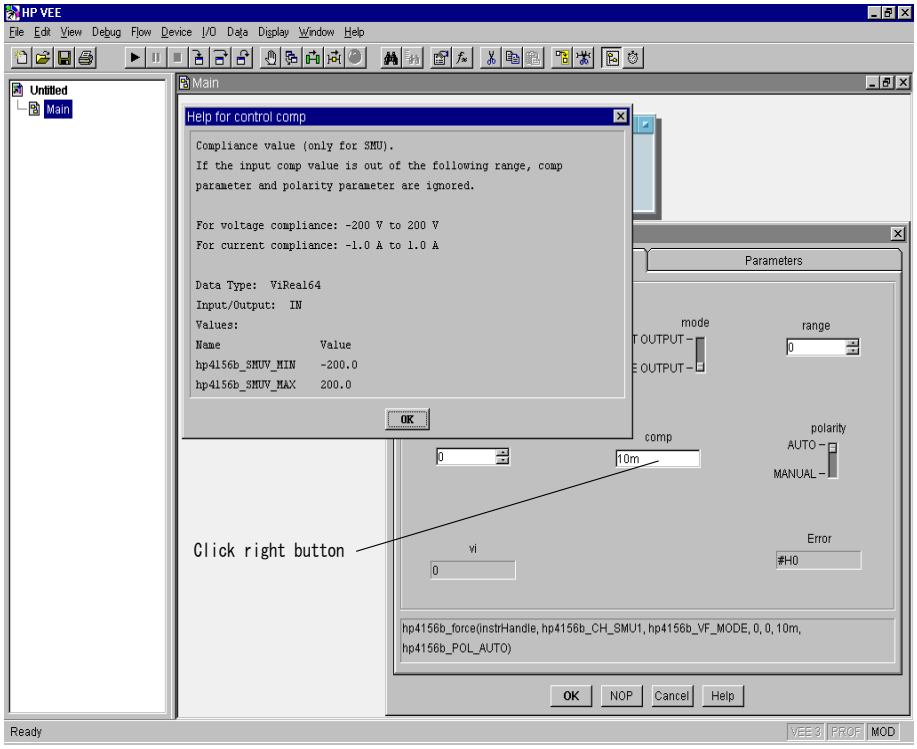
**Figure 6-5** **To Set Input Parameters**



## To use the Help function

If you need to know the details for each parameter in order to enter the parameter value, move the mouse pointer to the appropriate entry field, then click the right mouse button. The context-based Help function will be displayed. Figure 6-6 shows the Help message for the *comp* entry field.

**Figure 6-6** **Help Function**



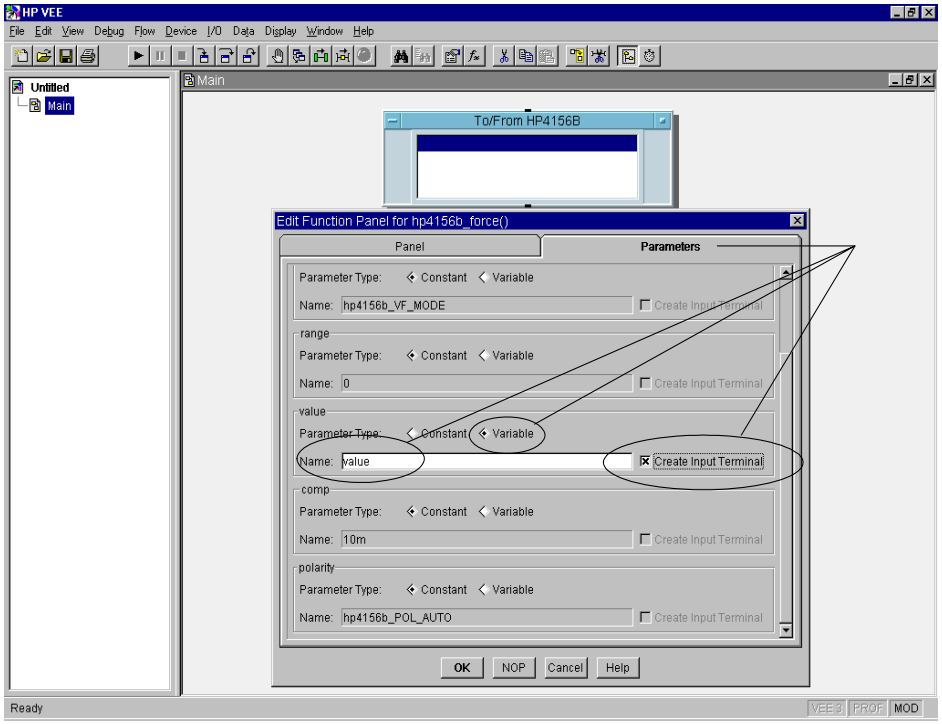

**NOTE** To open on-line Help for the *plug&play* driver, click the right mouse button in the To/From object, then select “Instrument Help”.

## To use input variables

Most of the Edit Function Panel dialog boxes have two tabs, Panel and Parameters. To change the value, enter the value in the Panel tab.

If you pass the value from another object, such as Data-Real object, click the Parameters tab, and use Variable (not Constant). See Figure 6-7.

**Figure 6-7** **To Use Input Variables**




**NOTE**

You can add terminals, after closing the dialog box, by placing the mouse pointer on the terminal area in the object and pressing Ctrl-A. You can also delete terminals by placing the mouse pointer on the terminal name you want to delete, and pressing Ctrl-D.

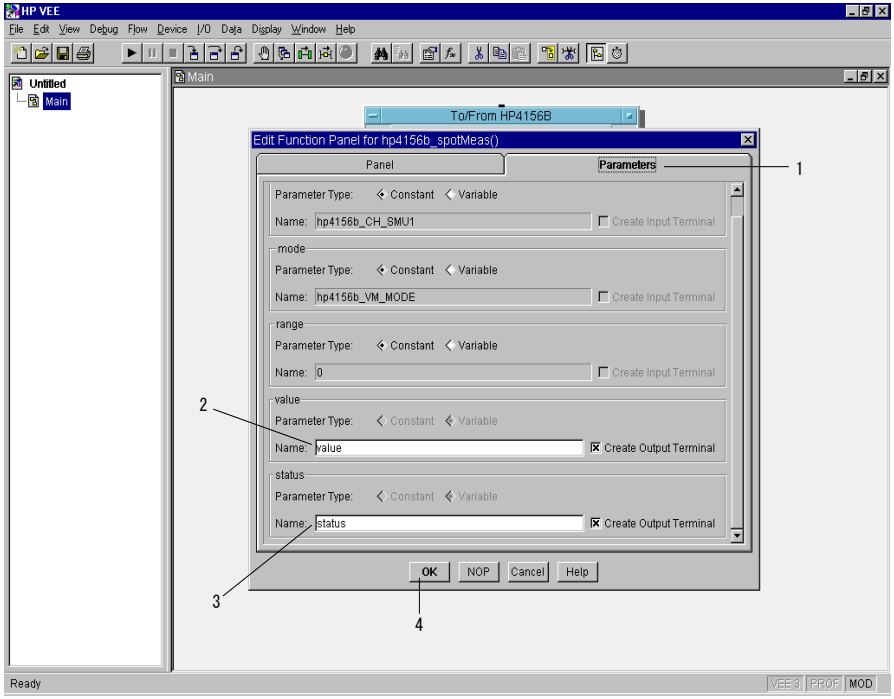
## To create output terminals in the To/From object

Figure 6-8 shows the Edit Function Panel dialog box of the hp4156b_spotMeas function. These measurement transactions need the output terminals in the To/From object. You can create the output terminal as shown in the following example.

1. Click the Parameters tab.
2. Enter the Name (output terminal name) for the output variable *value*.
3. Enter the Name (output terminal name) for the output variable *status*.
4. Click OK. The dialog box is closed, the transaction is added to the To/From object, and the output terminals are created in the object.

The output terminal will be created with the default name if steps 2 and 3 are omitted.

**Figure 6-8** **To Create Output Terminals**



## To display/connect the Data object

In Figure 6-9, the Data-Real object is used to pass the input parameter value to the *value* input variable of the hp4156b_force transaction.

You can display the Data-Real object by clicking the Data menu, selecting Constant, and then selecting Real. To pass the value, connect the output terminal of the Data-Real object to the input terminal of the To/From HP4156B object.

**NOTE**

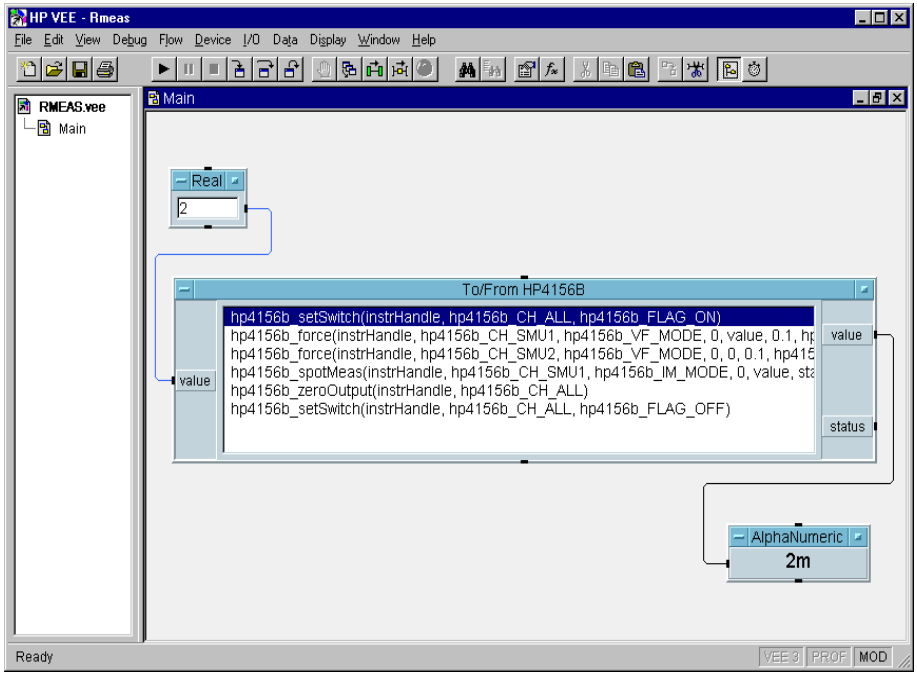
Confirm the data type of the input variable. The data type of the Data object must be the same as the data type for the input parameter.

## To display/connect the Display object

In Figure 6-9, the Display-AlphaNumeric object is used to display the measurement data (*result* output variable) from the hp4156b_spotMeas transaction.

You can display the Display-AlphaNumeric object by clicking the Display menu, and selecting AlphaNumeric. To display the value, connect the output terminal of the To/From HP4156B object to the input terminal of the Display-AlphaNumeric object.

**Figure 6-9** **To Connect Input/Output Terminals**



# Debugging Your Program

You may encounter problems when creating programs to control the 4155/4156. In the program development or debugging phase, insert the following transactions (functions of the driver) in the To/From object. Do not forget to remove the functions after completing the program. These functions will cause increased program execution time.

- hp4156b_cmd(instrHandle,”US”)
- hp4156b_errorQueryDetect

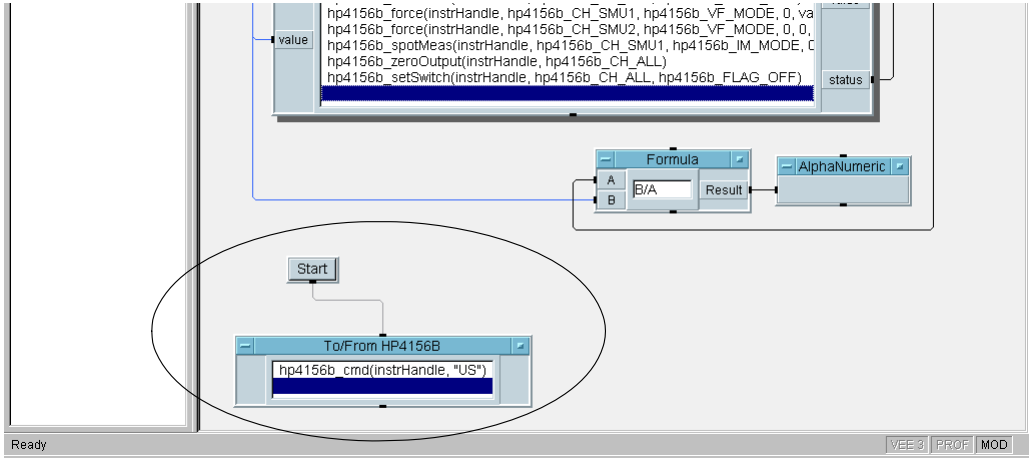
## To recover control mode

When using Agilent VEE, the 4155/4156 control mode is controlled by the hp4156b_init function, which is automatically called and executed by Agilent VEE when the program first runs after loading.

However, if you press any PAGE CONTROL key or LOCAL softkey on the 4155/4156 front panel after program execution, the control mode is changed. Also, if an unexpected I/O error has occurred, you may need to do a hardware reset which changes the control mode. Once the control mode is changed, the program cannot run without reloading it.

To recover the control mode without reloading the program, enter the US command using the hp4156b_cmd function as shown in Figure 6-10. The command recovers the effective control mode for the *plug&play* driver.

**Figure 6-10** **To Enter the US Command**



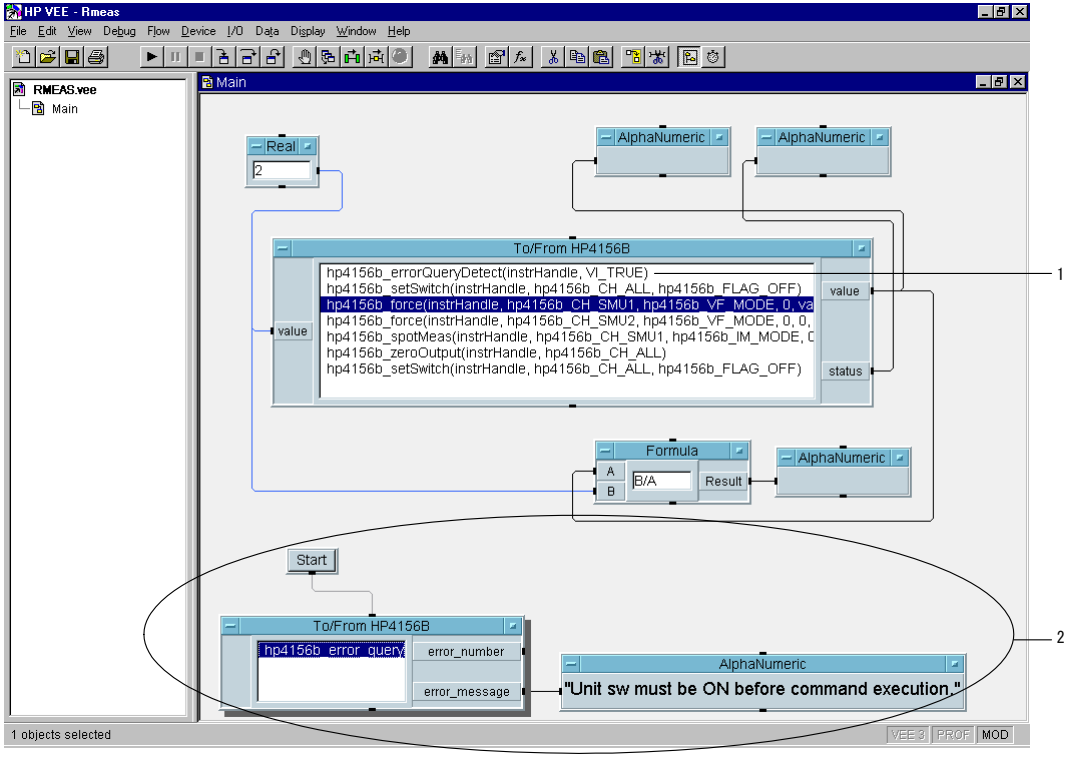
## To check for instrument error

The hp4156b_errorQueryDetect function enables or disables automatic instrument error checking. If automatic error checking is enabled, the driver will query the instrument for an error at the end of each function call.

If this function is enabled (1 of Figure 6-11) and if an error occurs in the function call, Agilent VEE stops the program execution and displays an error dialog box. You must then enter the hp4156b_error_query function (see 2 of Figure 6-11). The hp4156b_error_query function returns the instrument error code and error message.

In this example, an error occurred in the hp4156b_force function call. The cause of the error was an improper parameter setting for the hp4156b_setSwitch function.

**Figure 6-11 To Use the hp4156b_error_query Function**



# Restrictions When Using the Driver with Agilent VEE

When using Agilent VEE and any of the following functions for Agilent 4155/4156 and Agilent E5250A, certain restrictions will apply.

## Invalid functions in the VEE program

- hp4156b_init, hpe5250a_init

  Agilent VEE calls and executes these functions automatically when the program first runs after loading. These functions cannot be called in a program using Agilent VEE.

- hp4156b_close, hpe5250a_close

  Agilent VEE calls and executes these functions automatically when you close the program or Agilent VEE. These functions cannot be called in a program using Agilent VEE.

- hp4156b_error_message, hpe5250a_error_message

  These functions receive the error status of the plug&play driver function, and returns the error message. However, these functions are invalid in a program using Agilent VEE, because Agilent VEE does not pass the error status to the function.

## Invalid use of the NULL pointer

The measurement functions listed below allow you to use the NULL pointer to restrict the number of parameters returned from the function. However, the NULL pointer is not available for Agilent VEE programming.

- hp4156b_spotMeas
- hp4156b_sweepIv
- hp4156b_sweepMiv
- hp4156b_sweepPiv
- hp4156b_sweepPbias
- hp4156b_measureM
- hp4156b_measureP
- hp4156b_sample
- hp4156b_readData

# High-Speed Spot Measurements

To make high-speed spot measurements, use the following functions.

**Table 6-1 Functions for High-Speed Spot Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Executes measurement | hp4156b_spotMeas | channel,mode,range,value,status |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-13 on page 6-18. This program measures MOSFET drain current. The measurement setup is shown in Figure 6-12.

**Figure 6-12 Device Connection and Source Setup for Example Program**

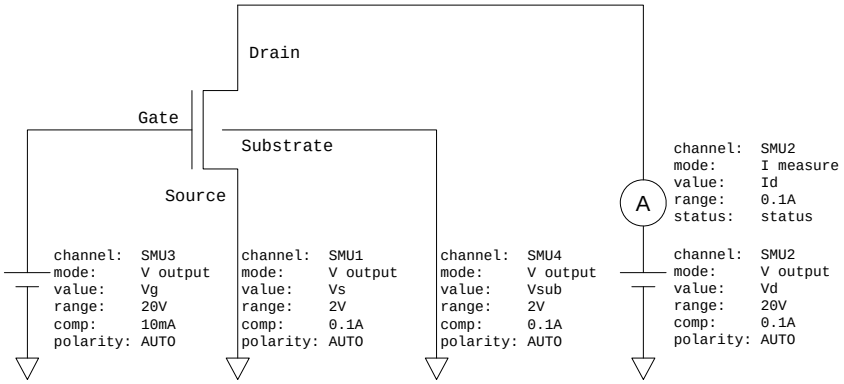


**Figure 6-13 Program Example of High-Speed Spot Measurement**

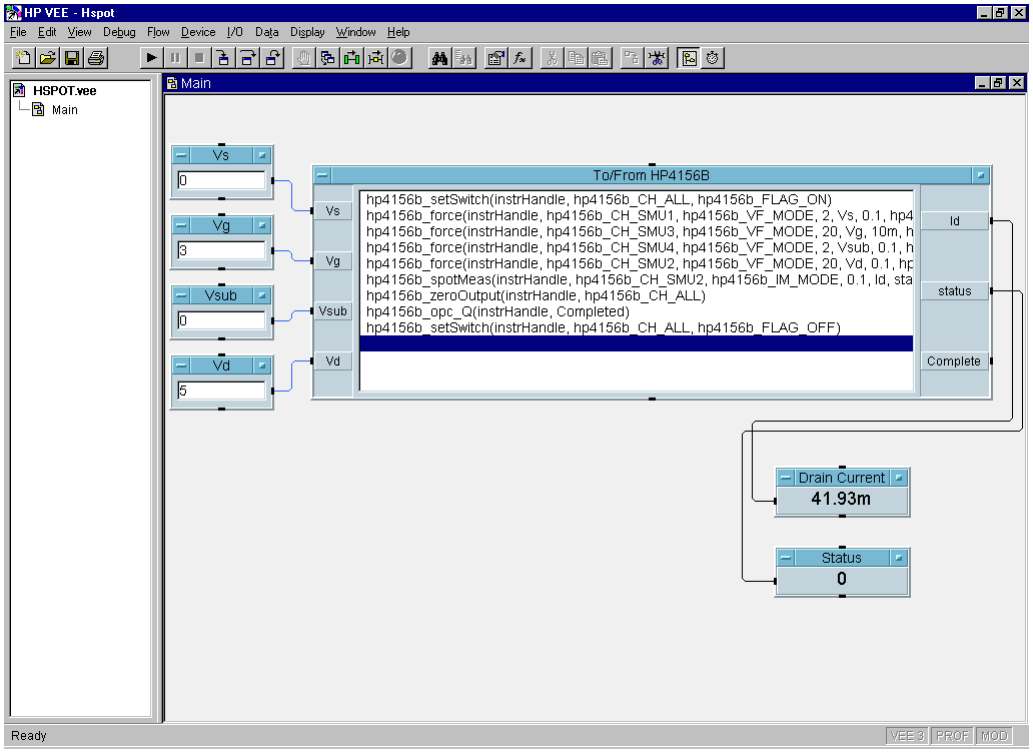



**Table 6-2 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vs,Vg,Vsub,Vd | Data-Constant-Real | Enters input parameters of hp4156b_force. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement. |
| Drain Current | Display-AlphaNumeric | Displays Id. (hp4156b_spotMeas value parameter) |
| Status | Display-AlphaNumeric | Displays status. (hp4156b_spotMeas status parameter) |

# Multi-Channel Spot Measurements

To make multi-channel spot measurements, use the following functions.

**Table 6-3 Functions for Multi-Channel Spot Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Executes measurement | hp4156b_measureM | channel[ ],mode[ ],range[ ],value[ ],status[ ] |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-15 on page 6-20. This program measures bipolar transistor collector current and base current. The example uses the User object of the Agilent VEE. See Figure 6-16 on page 6-21. The measurement setup is shown in Figure 6-14.

**Figure 6-14 Device Connection and Source Setup for Example Program**

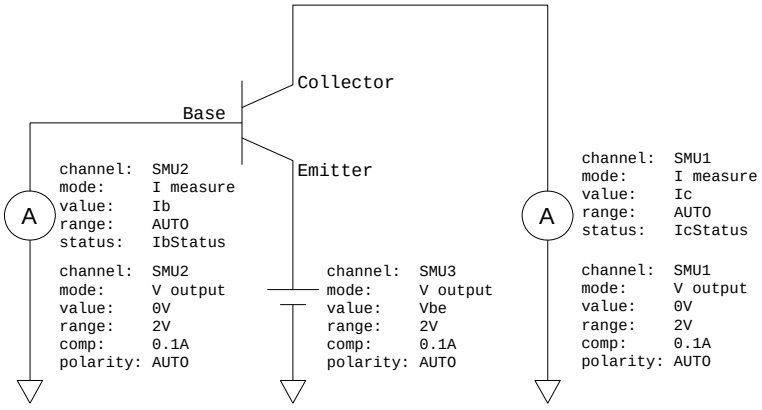


**Figure 6-15 Program Example of Multi-Channel Spot Measurement**

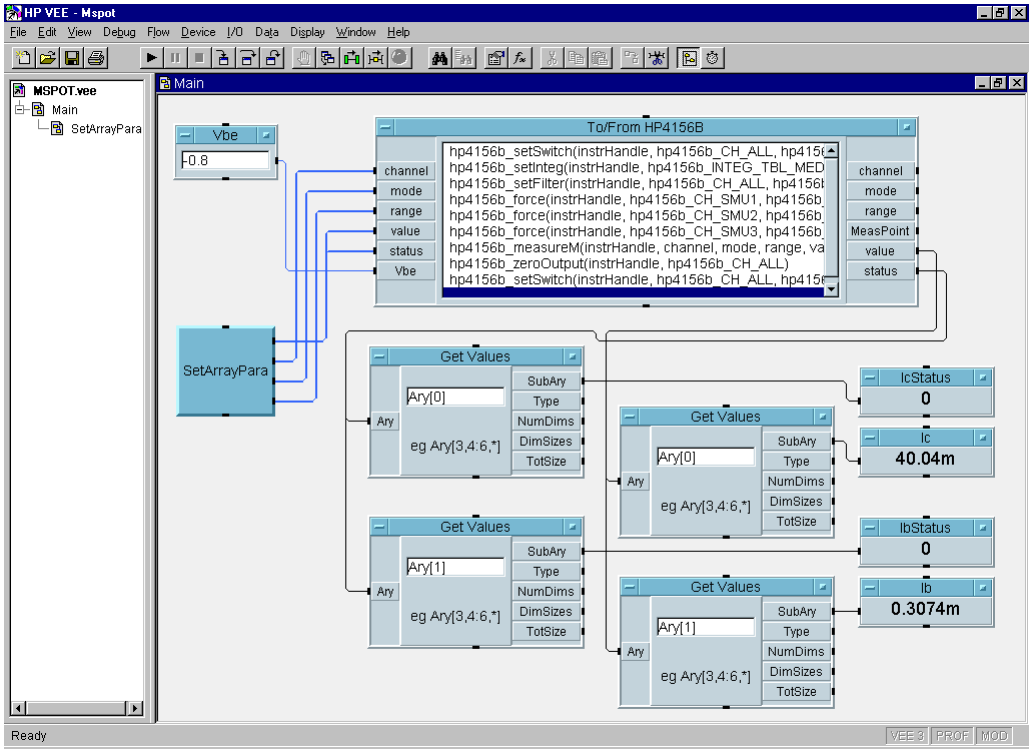



**Table 6-4 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vbe | Data-Constant-Real | Enters input parameters of hp4156b_force. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement. |
| GetValues | Data-AccessArray-GetValues | Gets data from Array (value[ ], status[ ]). |
| IcStatus,Ic, IbStatus,Ib | Display-AlphaNumeric | Displays measurement data/status. (hp4156b_measureM output parameters) |

**Figure 6-16 SetArrayPara User Object**

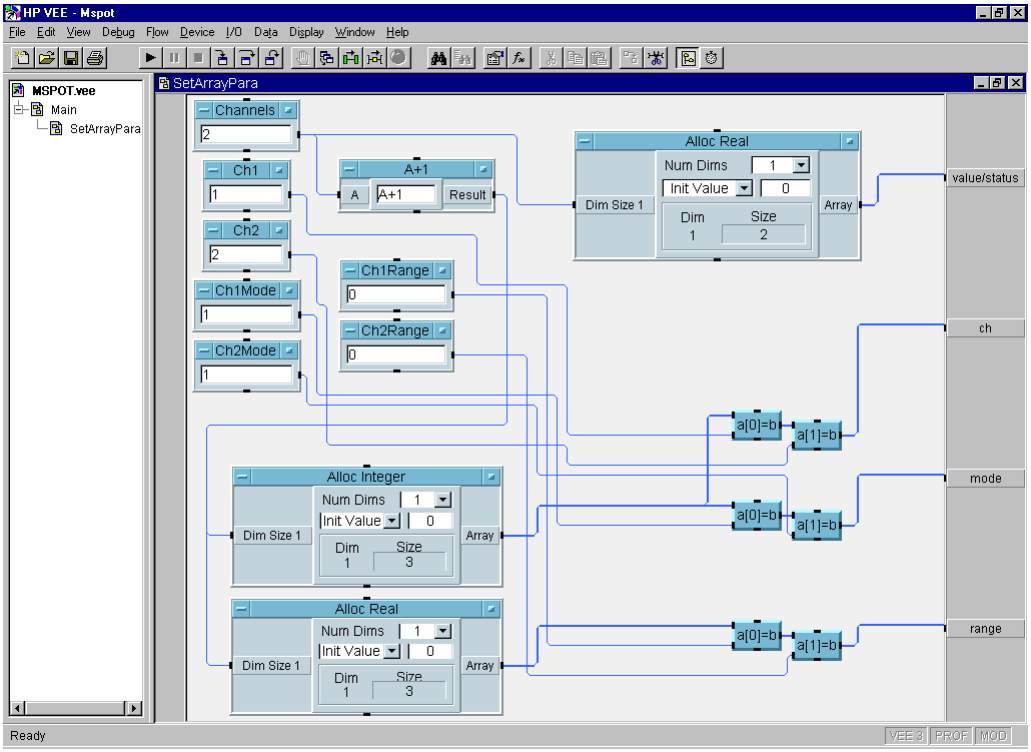



**Table 6-5 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Channels,Ch1,Ch2, Ch1Mode,Ch2Mode, Ch1Range,Ch2Range | Data-Constant-Integer Data-Constant-Real | Enters data to allocate array, and array element for channel[ ], mode[ ], range[ ] of hp4156b_measureM. |
| A+1 | Device-Formula | Calculates A+1 to allocate array. |
| AllocReal, AllocInteger | Data-AllocateArray-Real Data-AllocateArray-Integer | Allocates array for channel[ ], mode[ ], range[ ] of hp4156b_measureM. |
| a[0]=b, a[1]=b | Data-AccessArray-SetValues | Sets data of array (array element). |

# Staircase Sweep Measurements

To make staircase sweep measurements, use the following functions.

**Table 6-6 Functions for Staircase Sweep Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Sweep Source Setup | hp4156b_setIv | channel,mode,range,start,stop,point,hold,delay, s_delay,comp,p_comp |
| Executes measurement | hp4156b_sweepIv | channel,mode,range,point,source[ ],value[ ], status[ ] |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-18 on page 6-23. This program measures MOSFET Id-Vd characteristics. The measurement setup is shown in Figure 6-17.

**Figure 6-17 Device Connection and Source Setup for Example Program**

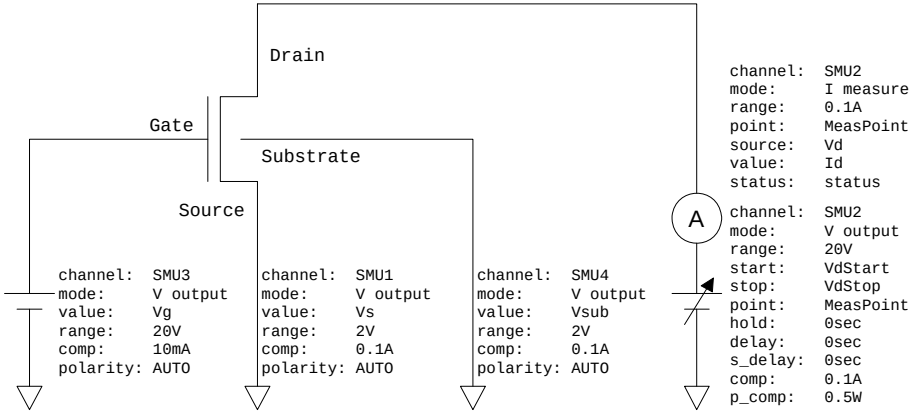


**Figure 6-18 Program Example of Staircase Sweep Measurement**

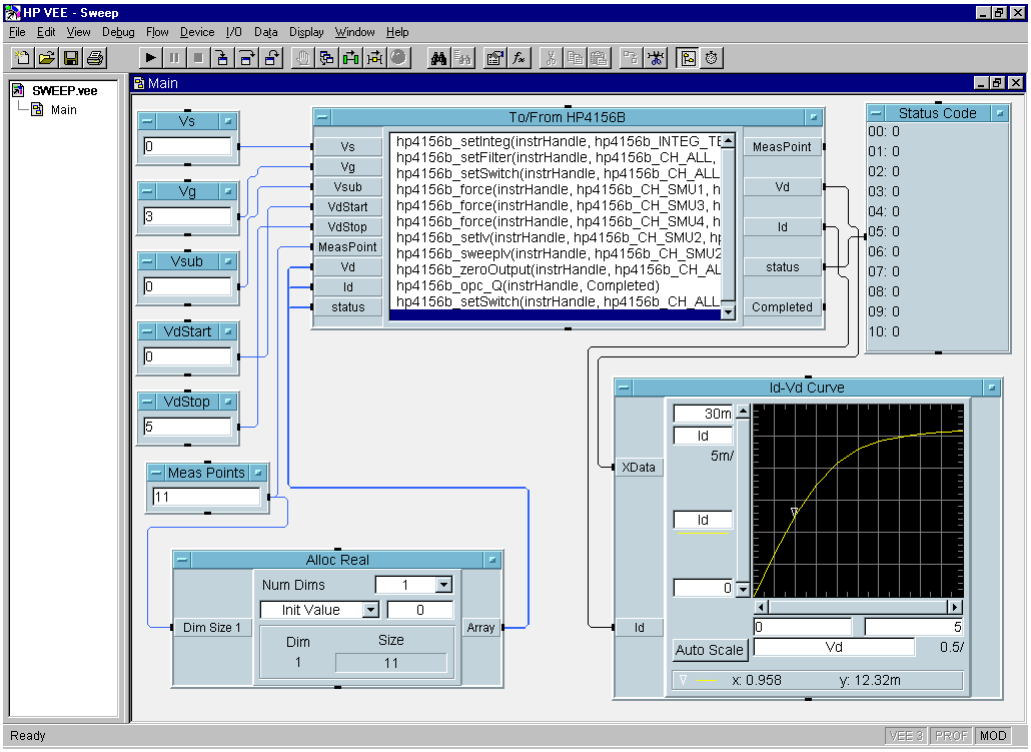



**Table 6-7 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vs,Vg,Vsub VdStart,VdStop MeasPoints | Data-Constant-Real Data-Constant-Integer | Enters input parameters of hp4156b_force, hp4156b_setIv, and hp4156b_sweepIv. |
| To/From HP4156B | I/O-InstrumentManager- Plug&play | Executes measurement. |
| AllocReal | Data-AllocateArray-Real | Allocates array for Vd[ ],Id[ ],status[ ] of hp4156b_sweepIv. |
| Status Code | Display-AlphaNumeric | Displays status[ ]. |
| Id-Vd Curve | Display-XvsYPlot | Plots Id-Vd curve. |

# Synchronous Sweep Measurements

To make synchronous sweep measurements, use the following function with the functions shown in “Staircase Sweep Measurements” on page 22, or “Pulsed Sweep Measurements” on page 34.

The hp4156b_setSweepSync function must be placed after the hp4156b_setIv function or the hp4156b_setPiv function in the To/From object.

**Table 6-8 Function for Synchronous Sweep Measurements**

| Description | Function | Parameters |
|---|---|---|
| Synchronous Source Setup | hp4156b_setSweepSync | channel,mode,range,start,stop,comp, p_comp |

A program example is shown in Figure 6-20 on page 6-25. This program measures MOSFET Id-Vg characteristics. The measurement setup is shown in Figure 6-19.

**Figure 6-19 Device Connection and Source Setup for Example Program**

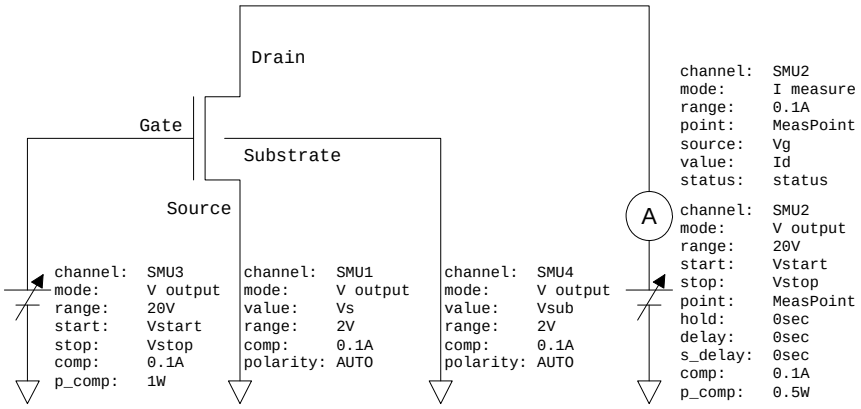


**Figure 6-20 Program Example of Synchronous Sweep Measurement**

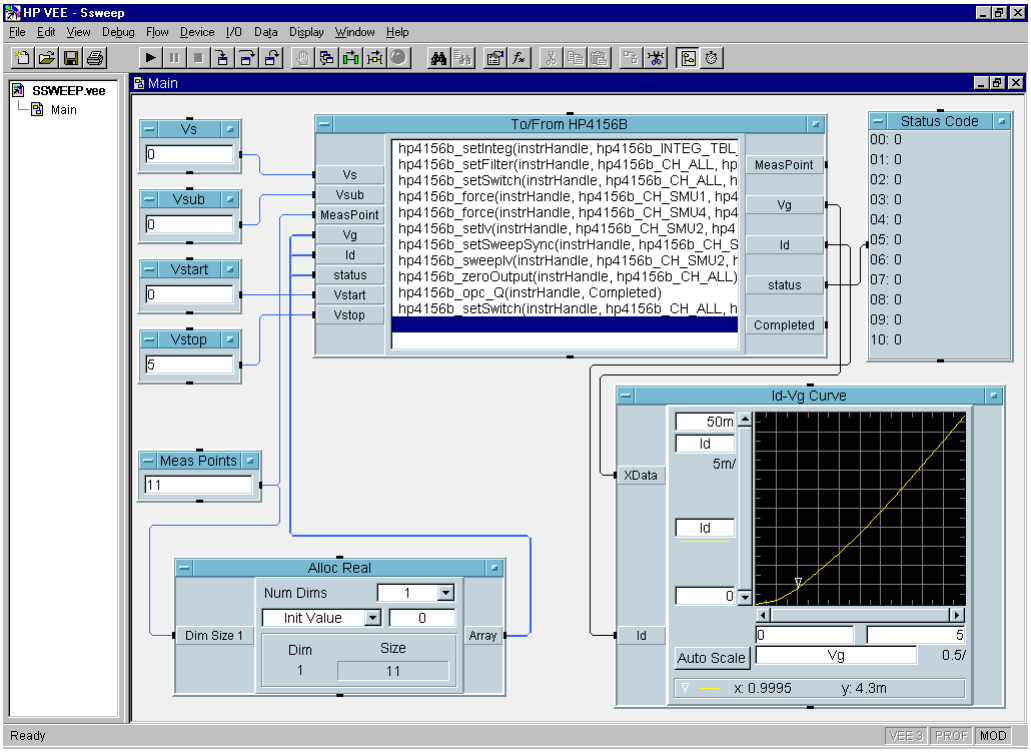



**Table 6-9 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vs,Vsub<br>Vstart,Vstop<br>MeasPoints | Data-Constant-Real<br>Data-Constant-Integer | Enters input parameters of hp4156b_force, hp4156b_setIv, and hp4156b_setSweepSync. |
| To/From HP4156B | I/O-InstrumentManager- Plug&play | Executes measurement. |
| AllocReal | Data-AllocateArray-Real | Allocates array for Vg[ ],Id[ ],status[ ] of hp4156b_sweepIv. |
| Status Code | Display-AlphaNumeric | Displays status[ ]. |
| Id-Vg Curve | Display-XvsYPlot | Plots Id-Vg curve. |

# Multi-Channel Sweep Measurements

To make multi-channel sweep measurements, use the following functions.

**Table 6-10 Functions for Multi-Channel Sweep Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Sweep Source Setup | hp4156b_setIv | channel,mode,range,start,stop,point,hold,delay, s_delay,comp,p_comp |
| Executes measurement | hp4156b_sweepMiv | channel[ ],mode[ ],range[ ],point,source[ ], value[ ], status[ ] |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-22 on page 6-27. This program measures bipolar transistor Ic, Ib-Vbe characteristics. The example uses the User Object of the Agilent VEE. See Figure 6-23 on page 6-28 and Figure 6-24 on page 6-29. The measurement setup is shown in Figure 6-21.

**Figure 6-21 Device Connection and Source Setup for Example Program**

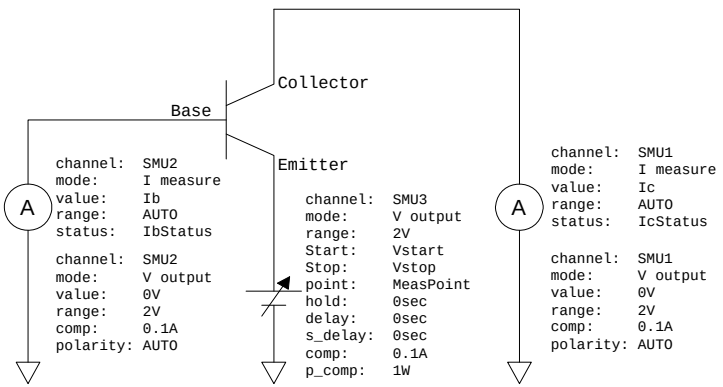


**Figure 6-22 Program Example of Multi-Channel Sweep Measurement**

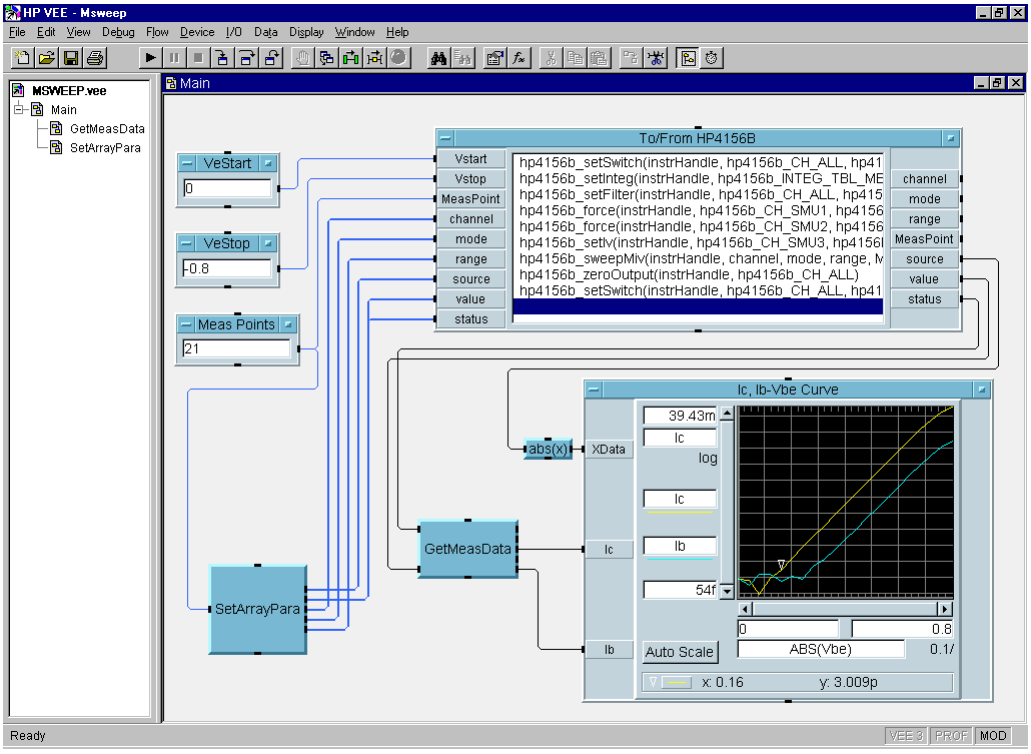



**Table 6-11 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| VeStart,VeStop<br>MeasPoints | Data-Constant-Real<br>Data-Constant-Integer | Enters input parameters of hp4156b_setIv and hp4156b_sweepMiv. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement. |
| abs(x) | Device-Formula | Calculates absolute value of Vbe (source). |
| Ic,Ib-Vbe Curve | Display-XvsYPlot | Plots Ic-Vbe and Ib-Vbe curves. |

**Figure 6-23 SetArrayPara User Object**

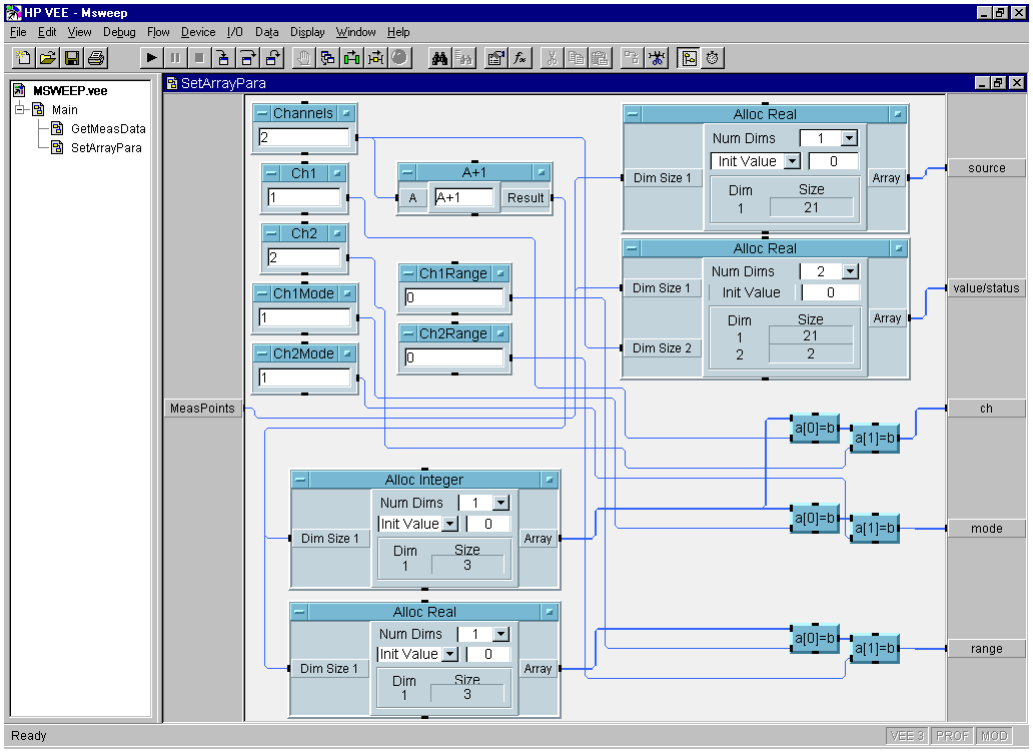



**Table 6-12 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Channels,Ch1,Ch2, Ch1Mode,Ch2Mode, Ch1Range,Ch2Range | Data-Constant-Integer<br>Data-Constant-Real | Enters data to allocate array, and array element for channel[ ], mode[ ], range[ ] of hp4156b_sweepMiv. |
| A+1 | Device-Formula | Calculates A+1 to allocate array. |
| AllocReal, AllocInteger | Data-AllocateArray-Real<br>Data-AllocateArray-Integer | Allocates array for channel, mode, range, source,value,status of hp4156b_sweepMiv. |
| a[0]=b, a[1]=b | Data-AccessArray-SetValues | Sets data of array (array element). |

**Figure 6-24 GetMeasData User Object**

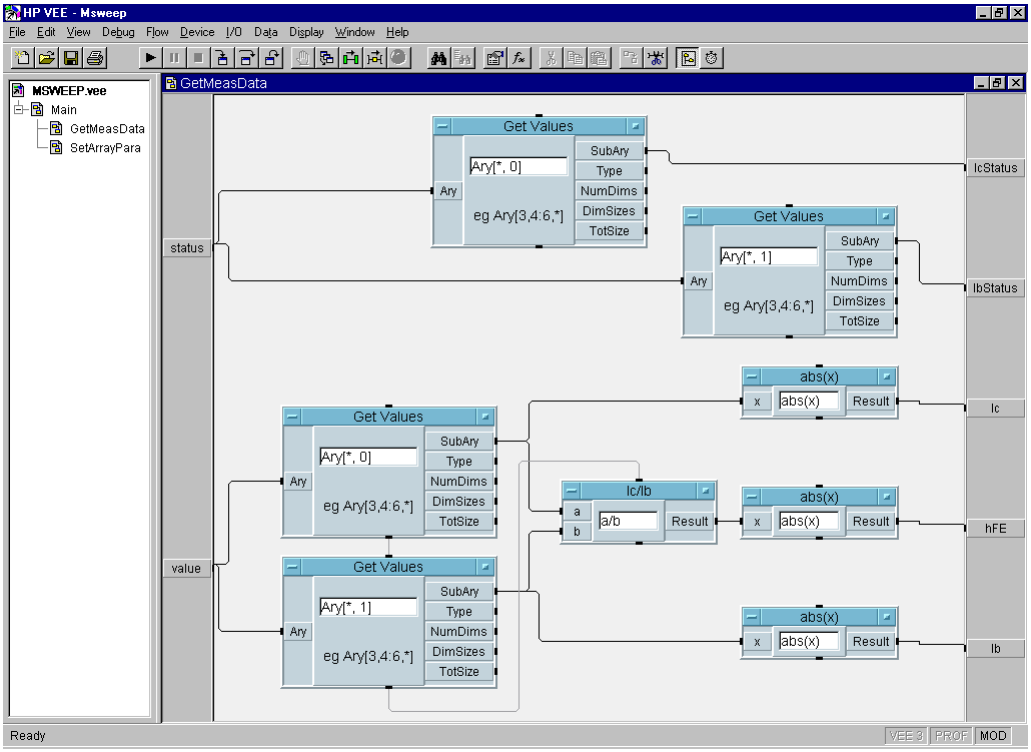



**Table 6-13 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| GetValues | Data-AccessArray-GetValues | Gets data from array (value[ ],status[ ]). |
| Ic/Ib | Device-Formula | Calculates a/b to get hFE value. |
| abs(x) | Device-Formula | Calculates absolute value of Ic,Ib,hFE. |

# Pulsed Spot Measurements

To make pulsed spot measurements, use the following functions.

**Table 6-14 Functions for Pulsed Spot Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state (pulse channel must be set to OFF) |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Forces pulse bias | hp4156b_setPbias | channel,mode,range,base,peak,width,period, hold,compliance |
| Executes measurement | hp4156b_measureP | channel,mode,range,value,status |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-26 on page 6-31. This program measures MOSFET drain current. The measurement setup is shown in Figure 6-25.

**Figure 6-25 Device Connection and Source Setup for Example Program**

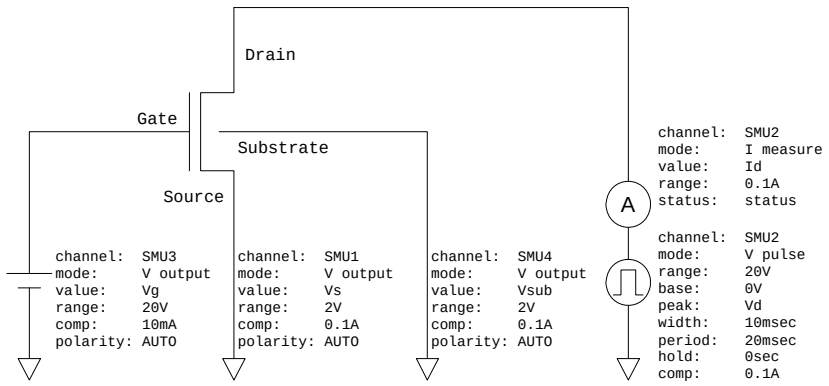


**Figure 6-26 Program Example of Pulsed Spot Measurement**

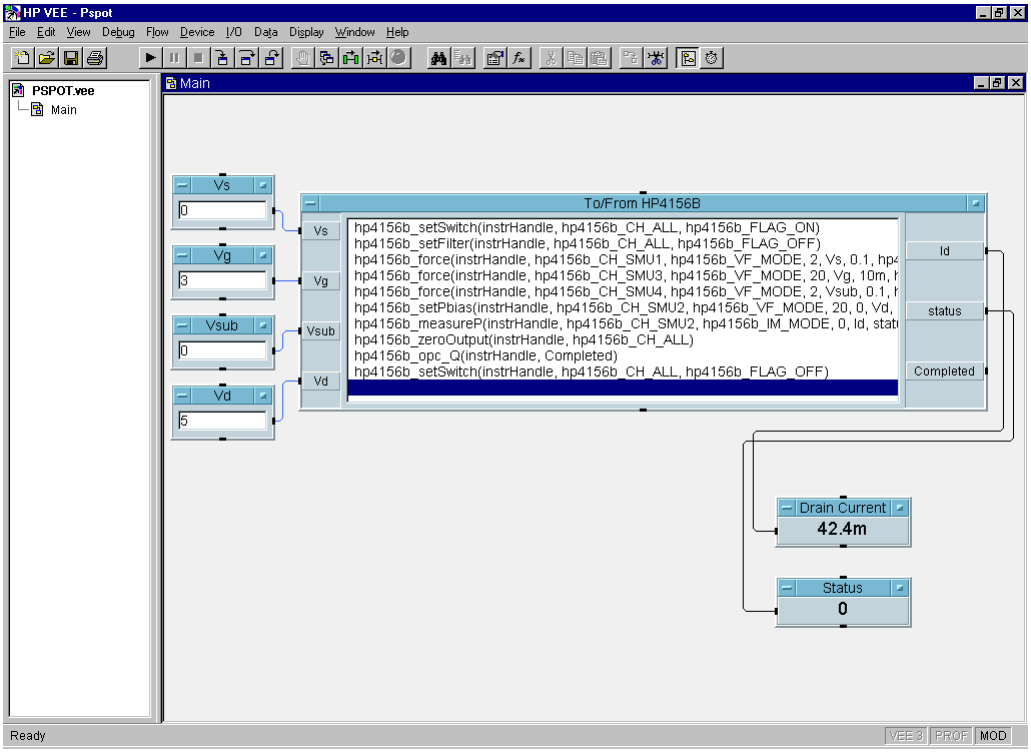



**Table 6-15 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vs,Vg,Vsub,Vd | Data-Constant-Real | Enters input parameters of hp4156b_force, and hp4156b_setPbias. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement. |
| Drain Current | Display-AlphaNumeric | Displays Id (hp4156b_measureP value parameter). |
| Status | Display-AlphaNumeric | Displays status (hp4156b_measureP status parameter). |

# Multi-Channel Pulsed Spot Measurements

To make multi-channel pulsed spot measurements, use the following functions.

**Table 6-16 Functions for Multi-Channel Pulsed Spot Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state (pulse channel must be set to OFF) |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Sends Command String | hp4156b_cmd | command (PT and PV commands are sent) |
| Executes measurement | hp4156b_startMeasure | meas_type,channel[ ],mode[ ],range[ ],source |
| Disables output | hp4156b_zeroOutput | channel |
| Reads measurement data | hp4156b_readData | eod,data_type,value,status,channel |

A program example is shown in Figure 6-28 on page 6-33. This program measures bipolar transistor collector current and base current. The measurement setup is shown in Figure 6-27.

**Figure 6-27 Device Connection and Source Setup for Example Program**

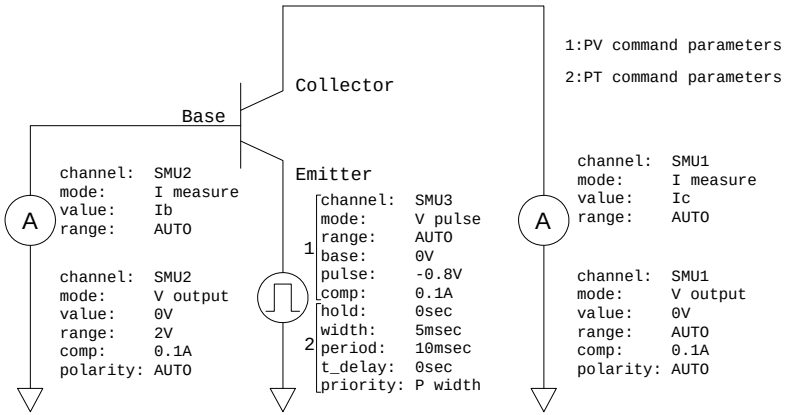


**Figure 6-28** **Program Example of Multi-Channel Pulsed Spot Measurement**

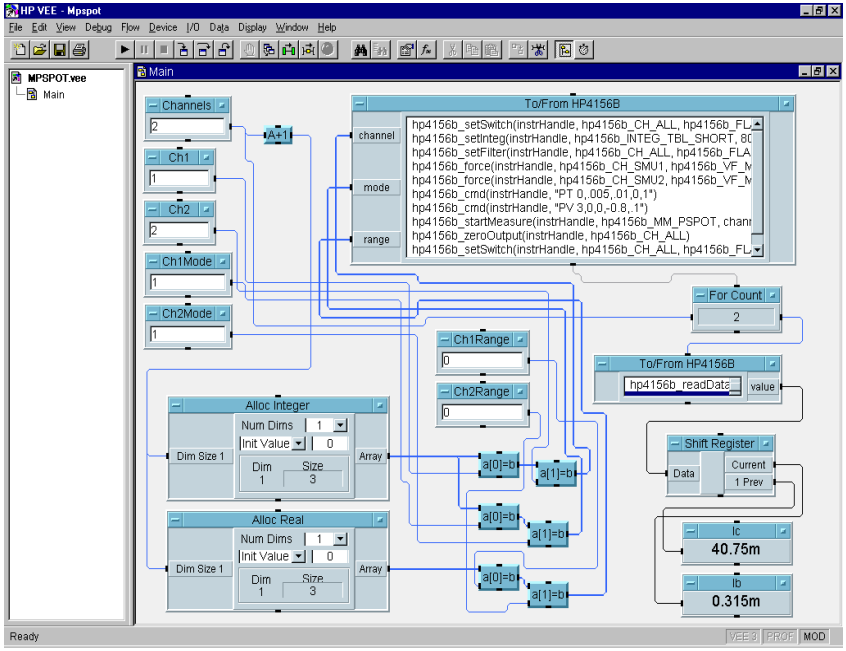



**Table 6-17** **Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Channels,Ch1,Ch2, Ch1Mode,Ch2Mode, Ch1Range,Ch2Range | Data-Constant-Integer<br>Data-Constant-Real | Enters data to allocate array, and array element for channel[ ], mode[ ], range[ ] of hp4156b_startMeasure. |
| A+1 | Device-Formula | Calculates A+1 to allocate array. |
| AllocReal, AllocInteger | Data-AllocateArray-Real<br>Data-AllocateArray-Integer | Allocates array for channel, mode, range of hp4156b_startMeasure. |
| a[0]=b, a[1]=b | Data-AccessArray-SetValues | Sets data of array (array element). |
| For Count | Flow-Repeat-ForCount | Repeats next action for specified count. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement or reads data. |
| Shift Register | Device-ShiftRegister | Outputs last data and 1 prev data. |
| Ic, Ib | Display-AlphaNumeric | Displays Ic and Ib. |

# Pulsed Sweep Measurements

To make pulsed sweep measurements, use the following functions.

**Table 6-18 Functions for Pulsed Sweep Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state (pulse channel must be set to OFF) |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Sweep Source Setup | hp4156b_setPiv | channel,mode,range,base,start,stop,point,hold, width,period,compliance |
| Executes measurement | hp4156b_sweepPiv | channel,mode,range,point,source[ ],value[ ], status[ ] |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-30 on page 6-35. This program measures MOSFET Id-Vd characteristics. The measurement setup is shown in Figure 6-29.

**Figure 6-29 Device Connection and Source Setup for Example Program**

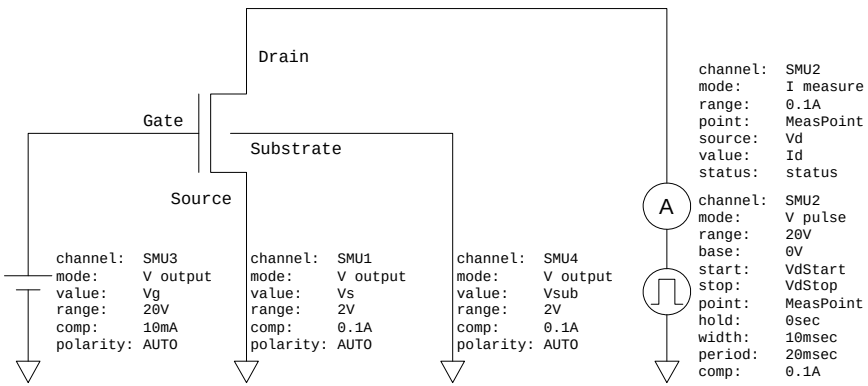


**Figure 6-30 Program Example of Pulsed Sweep Measurement**

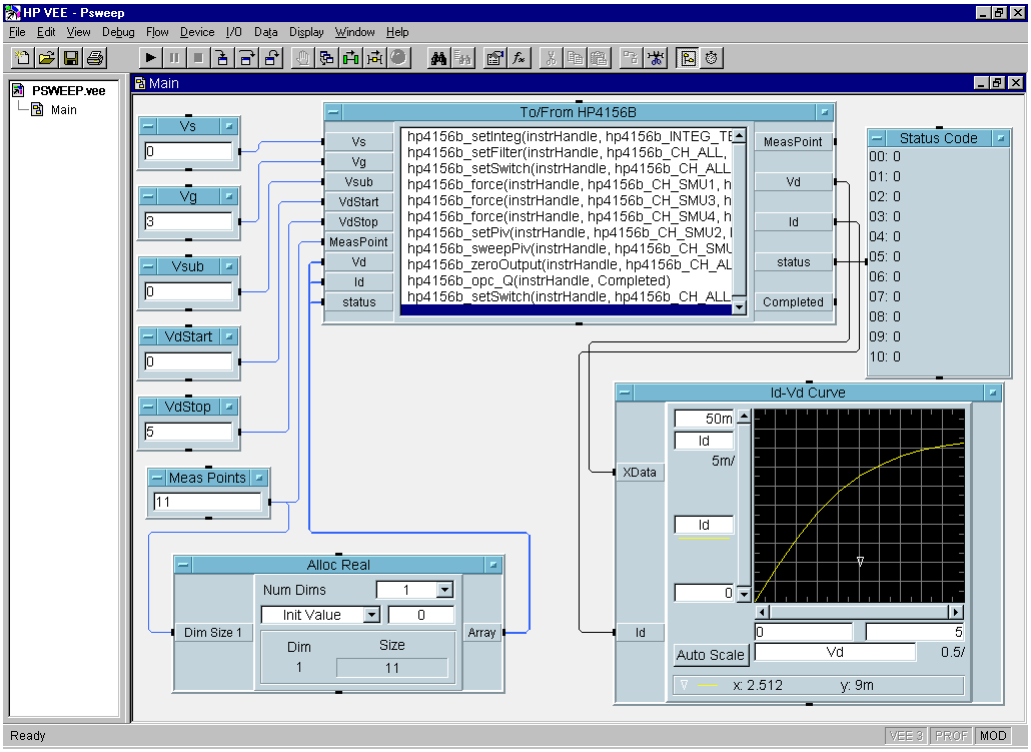



**Table 6-19 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vs,Vg,Vsub VdStart,VdStop MeasPoints | Data-Constant-Real Data-Constant-Integer | Enters input parameters of hp4156b_force, hp4156b_setPiv, and hp4156b_sweepPiv. |
| To/From HP4156B | I/O-InstrumentManager- Plug&play | Executes measurement. |
| AllocReal | Data-AllocateArray-Real | Allocates array for Vd[ ],Id[ ],status[ ] of hp4156b_sweepIv. |
| Status Code | Display-AlphaNumeric | Displays status[ ]. |
| Id-Vd Curve | Display-XvsYPlot | Plots Id-Vd curve. |

# Multi-Channel Pulsed Sweep Measurements

To make multi-channel pulsed sweep measurements, use the following functions.

**Table 6-20 Functions for Multi-Channel Pulsed Sweep Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state (pulse channel must be set to OFF) |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Sends Command String | hp4156b_cmd | command (PT and PWV commands are sent) |
| Executes measurement | hp4156b_startMeasure | meas_type,channel[ ],mode[ ],range[ ],source |
| Disables output | hp4156b_zeroOutput | channel |
| Reads measurement data | hp4156b_readData | eod,data_type,value,status,channel |

A program example is shown in Figure 6-32 on page 6-37. This program measures bipolar transistor Ic, Ib-Vbe characteristics. The example uses the User Function of the Agilent VEE. See Figure 6-33 on page 6-38. The measurement setup is shown in Figure 6-31.

**Figure 6-31 Device Connection and Source Setup for Example Program**

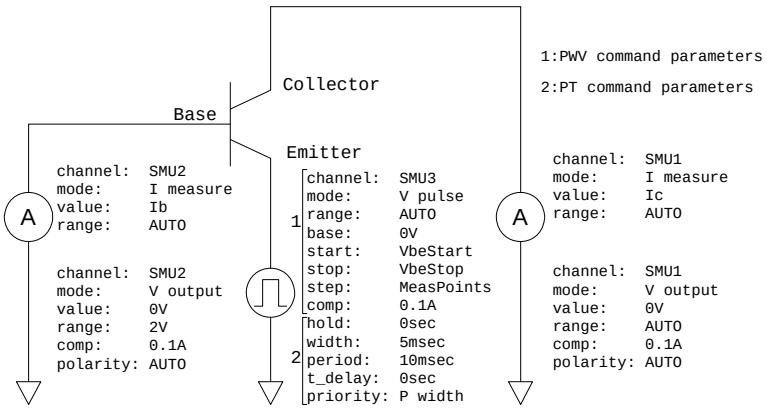


**Figure 6-32 Program Example of Multi-Channel Pulsed Sweep Measurement**

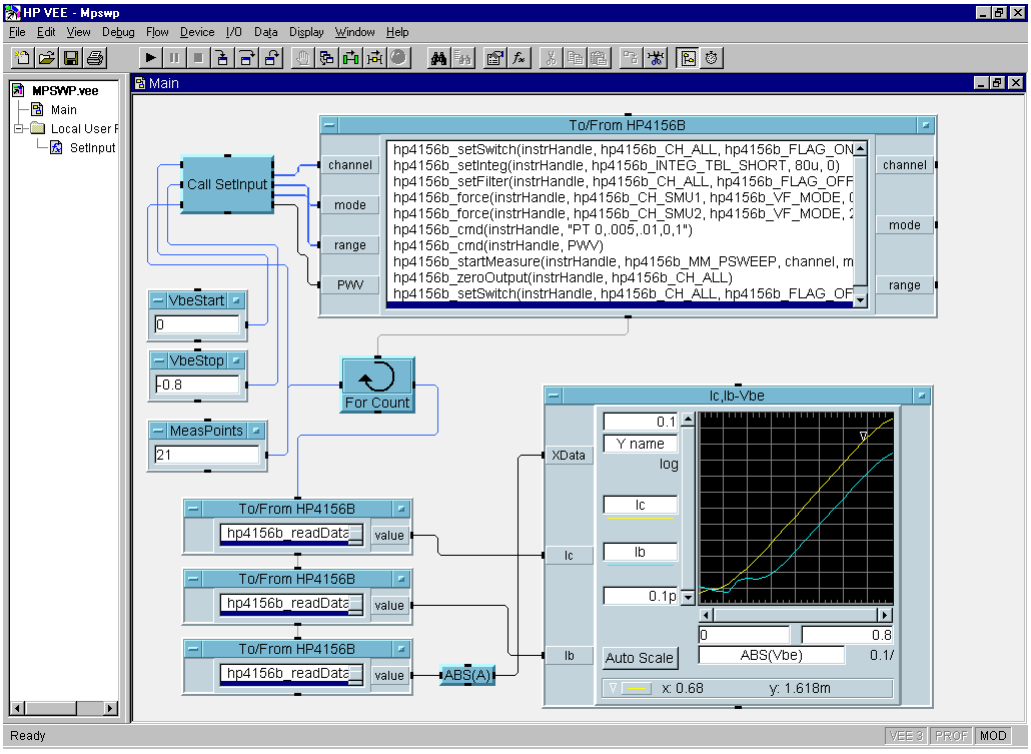



**Table 6-21 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| VbeStart,VbeStop<br>MeasPoints | Data-Constant-Real<br>Data-Constant-Integer | Enters PWV command parameters. |
| For Count | Flow-Repeat-ForCount | Repeats next action for specified count. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement, or reads data. |
| ABS(A) | Device-Formula | Calculates absolute value of Vbe (source). |
| Ic,Ib-Vbe | Display-XvsYPlot | Plots Ic-Vbe and Ib-Vbe curves. |

**Figure 6-33 SetInput User Function**

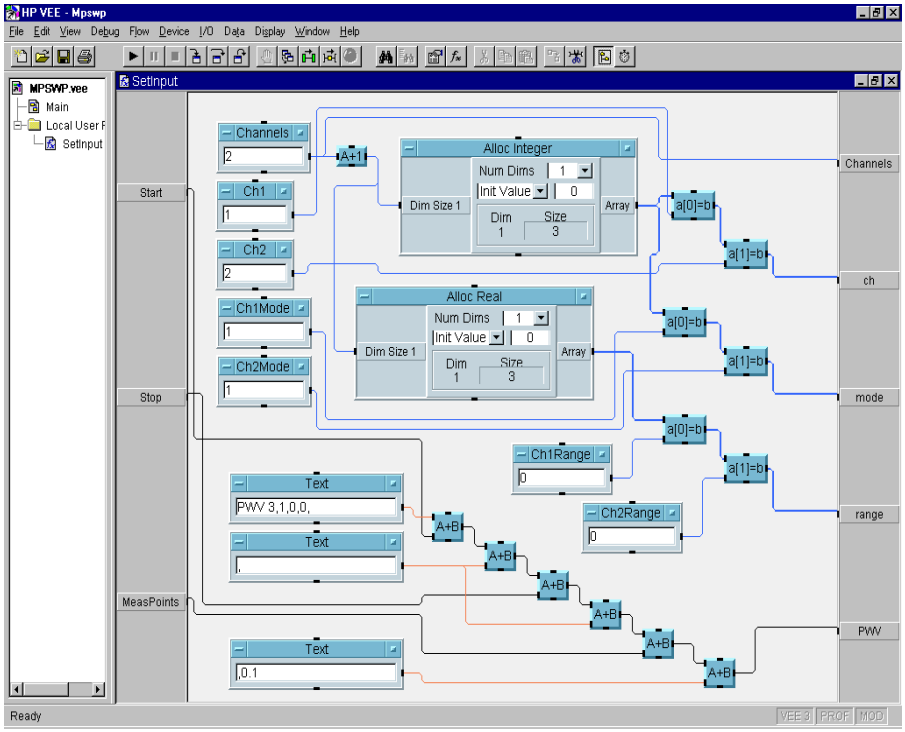



**Table 6-22 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Channels,Ch1,Ch2, Ch1Mode,Ch2Mode, Ch1Range,Ch2Range | Data-Constant-Integer Data-Constant-Real | Enters data to allocate array, and array element for channel[ ], mode[ ], range[ ] of hp4156b_startMeasure. |
| A+1 | Device-Formula | Calculates A+1 to allocate array. |
| AllocReal, AllocInteger | Data-AllocateArray-Real Data-AllocateArray-Integer | Allocates array for channel, mode, range, of hp4156b_startMeasure. |
| a[0]=b, a[1]=b | Data-AccessArray-SetValues | Sets data of array (array element). |
| Text | Data-Constant-Text | Enters PWV command parameters. |
| A+B | Device-Formula | Calculates A+B to create PWV command. |

# Staircase Sweep with Pulsed Bias Measurements

For staircase sweep with pulsed bias measurements, use the following functions.

**Table 6-23 Functions for Staircase Sweep with Pulsed Bias Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state (pulse channel must be set to OFF) |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance,polarity |
| Forces pulse bias | hp4156b_setPbias | channel,mode,range,base,peak,width,period, hold,compliance |
| Sweep Source Setup | hp4156b_setIv | channel,mode,range,start,stop,point,hold,delay, s_delay,comp,p_comp |
| Executes measurement | hp4156b_sweepPbias | channel,mode,range,point,source[ ],value[ ], status[ ] |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-35 on page 6-40. This program measures bipolar transistor Ic-Vc characteristics. The measurement setup is shown in Figure 6-34.

**Figure 6-34 Device Connection and Source Setup for Example Program**

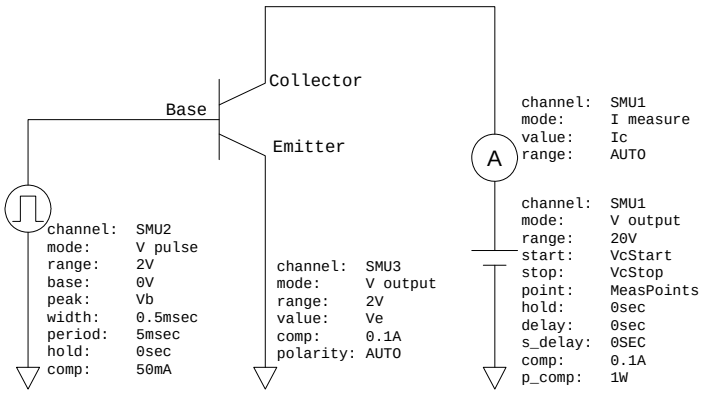


**Figure 6-35 Program Example of Staircase Sweep with Pulsed Bias Measurement**

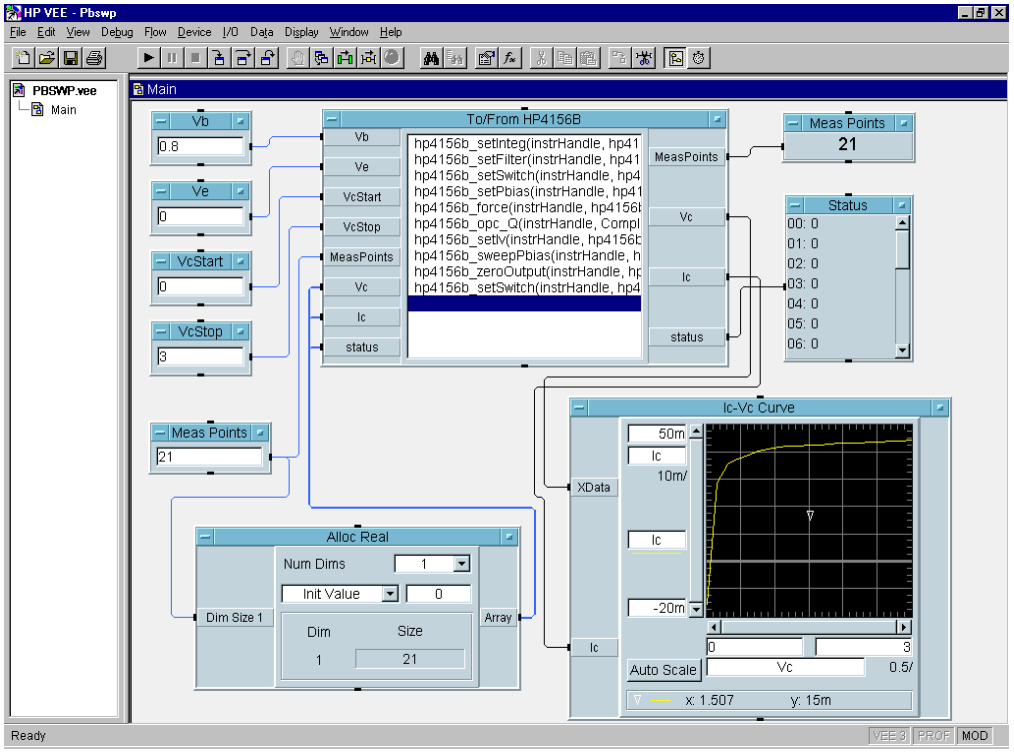



**Table 6-24 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| VcStart, VcStop Vb,Ve,MeasPoints | Data-Constant-Real Data-Constant-Integer | Enters input parameters of hp4156b_force hp4156b_setPbias, and hp4156b_setIv. |
| To/From HP4156B | I/O-InstrumentManager- Plug&play | Executes measurement. |
| AllocReal | Data-AllocateArray-Real | Allocates array for Vc[ ],Ic[ ],status[ ] of hp4156b_sweepPbias. |
| MeasPoints | Display-AlphaNumeric | Displays number of measurement points. |
| Status | Display-AlphaNumeric | Displays status[ ]. |
| Ic-Vc Curve | Display-XvsYPlot | Plots Ic-Vc curve. |

# Sampling Measurements

To make sampling measurements, use the following functions.

**Table 6-25 Functions for Sampling Measurements**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| Output Filter Setup | hp4156b_setFilter | channel,state |
| Integration Time Setup | hp4156b_setInteg | table,time,average |
| Forces dc bias | hp4156b_force | channel,mode,range,value, compliance,polarity |
| Sampling timing setup | hp4156b_setSample | hold,interval,point |
| Sampling dc source setup | hp4156b_addSampleSyncIv | channel,mode,range,base,bias, compliance |
| Sampling pulse source setup | hp4156b_addSampleSyncPulse | channel,base,peak,count,period, width,delay,rise,fall |
| Executes measurement | hp4156b_sample | channel[ ],mode[ ],range[ ],point, index[ ],value[ ],status[ ] |
| Clears sampling source setup | hp4156b_clearSampleSync | |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-37. This program measures resistance. The example uses the User function of the Agilent VEE. See Figure 6-38 on page 6-43 and Figure 6-39 on page 6-44. The measurement setup is shown in Figure 6-36.

**Figure 6-36 Device Connection and Source Setup for Example Program**

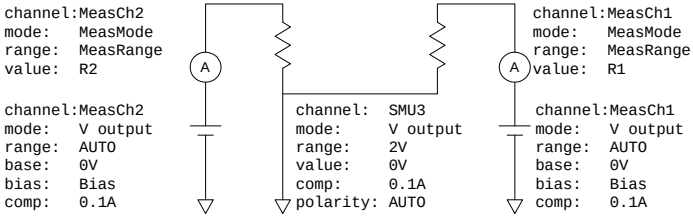


**Figure 6-37 Program Example of Sampling Measurement**

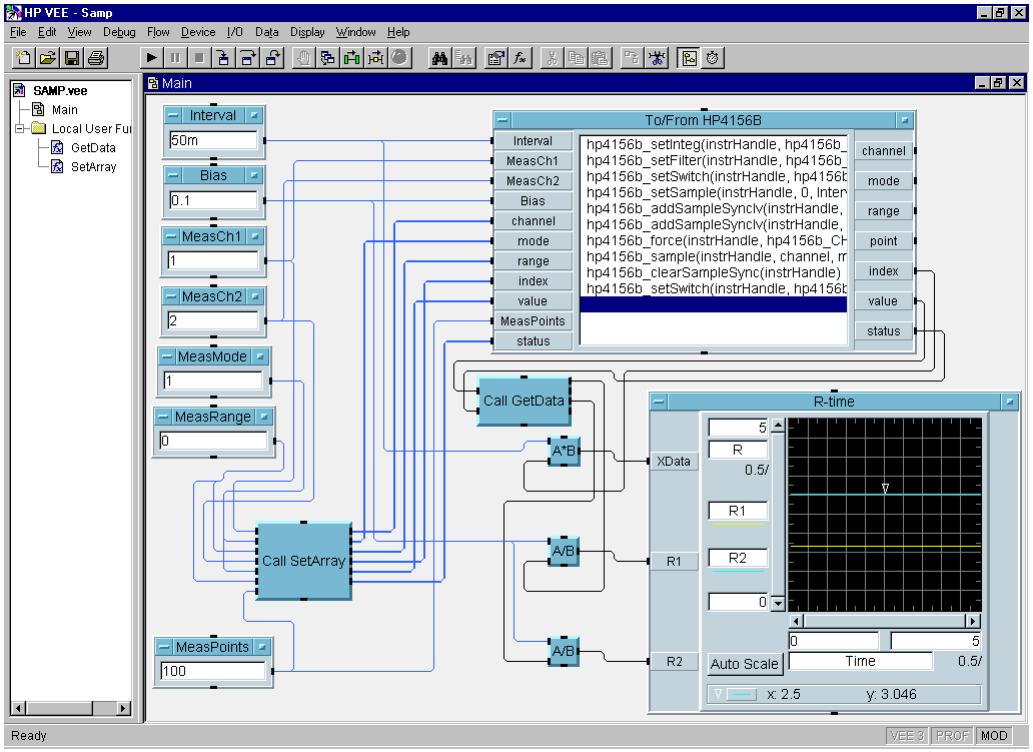



**Table 6-26 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Interval,Bias, MeasCh1,MeasCh2 MeasMode, MeasRange, MeasPoints | Data-Constant-Integer, Data-Constant-Real | Enters input parameters of hp4156b_force, hp4156b_setSample, hp4156b_addSampleSyncIv, hp4156b_addSampleSyncPulse, hp4156b_sample |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Executes measurement. |
| A*B | Device-Formula | Calculates A*B to get Time value (X). |
| A/B | Device-Formula | Calculates A/B to get R1, R2 value (Y). |
| R-time | Display-XvsYPlot | Plots R-t curves. |

**Figure 6-38 SetArray User Function**

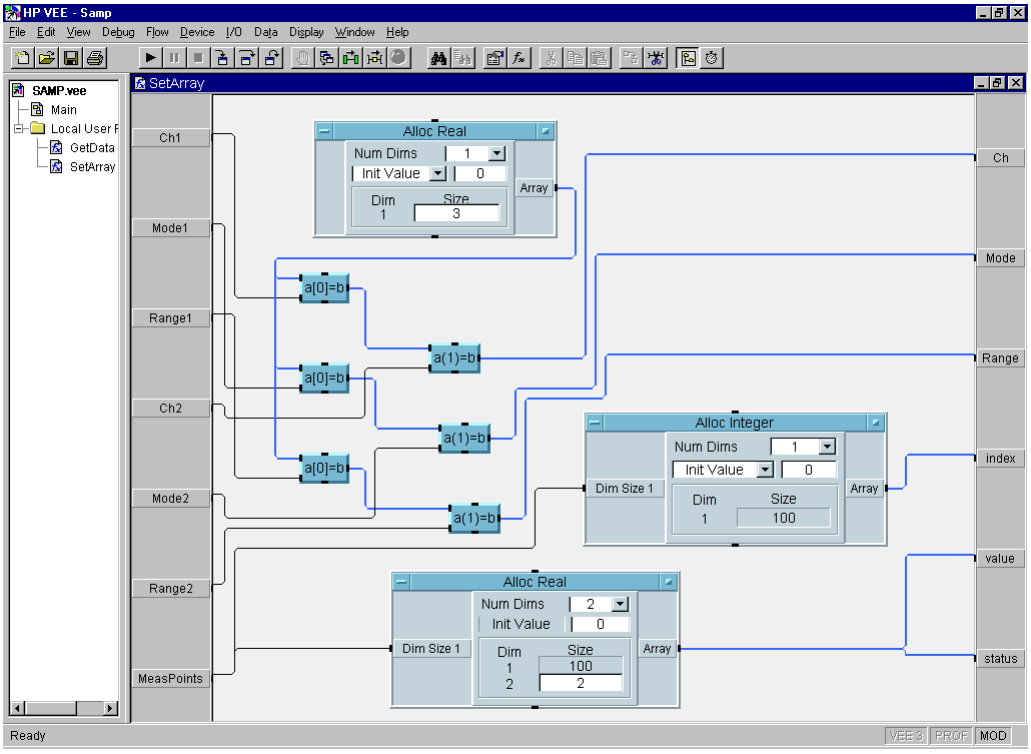



**Table 6-27 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| AllocReal, AllocInteger | Data-AllocateArray-Real Data-AllocateArray-Integer | Allocates array for channel, mode, range, index,value,status of hp4156b_sample. |
| a[0]=b, a[1]=b | Data-AccessArray-SetValues | Sets data of array (array element). |

**Figure 6-39 GetData User Function**

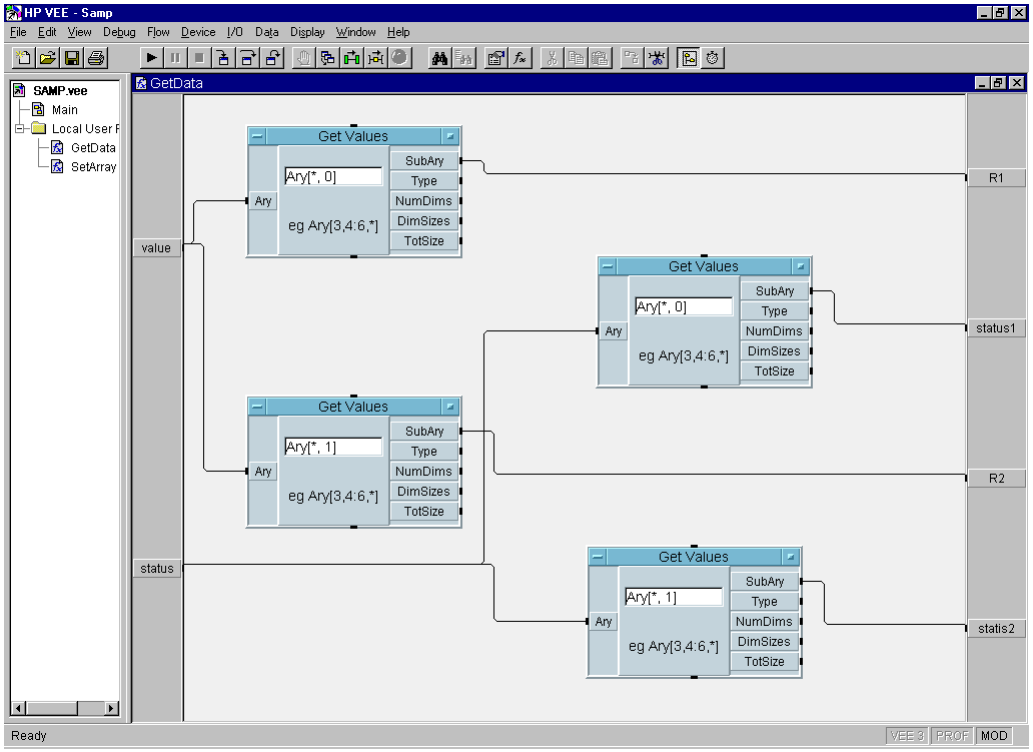



**Table 6-28 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| GetValues | Data-AccessArray-GetValues | Gets data from array (value[ ],status[ ]). |

# Stress Force

For stress force, use the following functions.

**Table 6-29 Functions for Stress Force**

| Description | Function | Parameters |
|---|---|---|
| Output Switch Setup | hp4156b_setSwitch | channel,state |
| PGU output impedance setup | hp4156b_setPguR | channel,state |
| Forces dc bias | hp4156b_force | channel,mode,range,value,compliance, polarity |
| Stress timing setup | hp4156b_setStress | hold,mode,duration,period |
| dc stress setup | hp4156b_addStressSyncIv | source,channel,mode,range,base,stress, compliance |
| Pulse stress setup | hp4156b_addStressSyncPulse | source,channel,base,stress,width,delay, rise,fall |
| Forces stress | hp4156b_stress | status |
| Clears stress source setup | hp4156b_clearStressSync | |
| Disables output | hp4156b_zeroOutput | channel |

A program example is shown in Figure 6-41 on page 6-46. This program forces dc stress and pulse stress to the DUTs (device under test). The measurement setup is shown in Figure 6-40.

**Figure 6-40 Device Connection and Source Setup for Example Program**

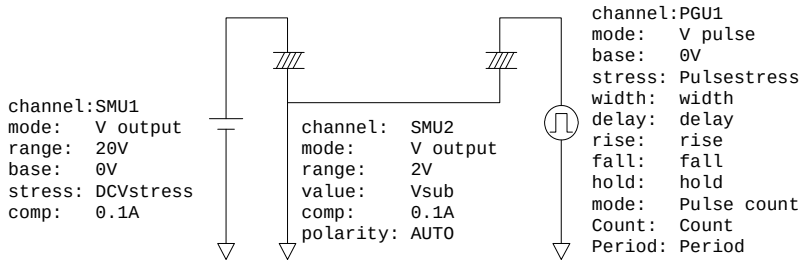


**Figure 6-41 Program Example of Stress Force**

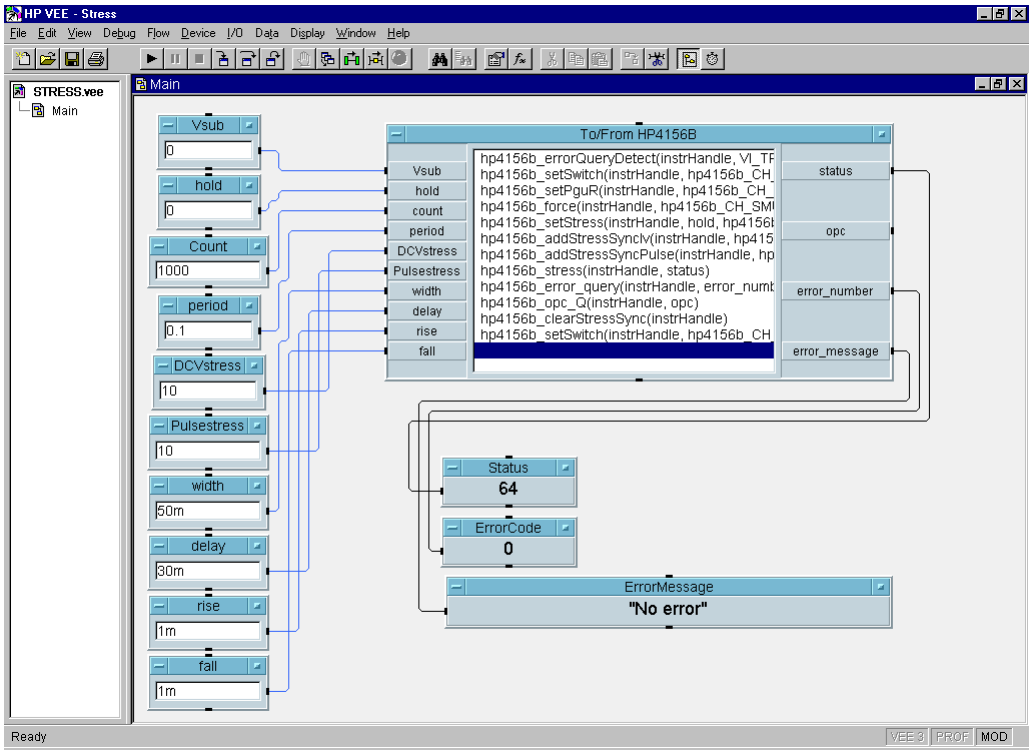



**Table 6-30 Program Explanation**

| Object Title | Menu | Explanation |
|---|---|---|
| Vsub,hold,Count, period,DCVstress, Pulsestress,width, delay,rise,fall | Data-Constant-Real<br>Data-Constant-Integer | Enters input parameters of hp4156b_force, hp4156b_setStress, hp4156b_addStressSyncIv, hp4156b_addStressSyncPulse, hp4156b_stress. |
| To/From HP4156B | I/O-InstrumentManager-Plug&play | Forces dc stress and pulse stress. |
| Status,ErrorCode, ErrorMessage | Display-AlphaNumeric | Displays measurement status, error code, error message. Ignore status code 64 which is meaningless as response of hp4156b_stress. |

# 7 Sample Application Programs Using VEE

This chapter explains how to use the sample application programs stored on the Agilent VEE Sample Program Disk furnished with Agilent 4155/4156. This chapter consists of the following sections:

- “Introduction”
- “Installation”
- “Using sample1.vee”
- “Using sample2.vee”
- “Customizing Sample Programs”

**CAUTION** The program and setup files stored on the Sample Program Disk are examples only, and may need to be customized for your specific application. Agilent Technologies is not responsible for any damage that may occur from the use of these sample programs.

**NOTE** Copy the Agilent VEE Sample Program Disk, and keep the original disk as a backup. To store a program after modifying it, use a file name that is different than the original program name.

# Introduction

This section introduces the sample application programs for Agilent VEE that are furnished with the 4155/4156, and covers the following sections:

- “Agilent VEE Sample Program Disk”
- “What are Sample Programs?”

## Agilent VEE Sample Program Disk

The Agilent VEE Sample Program Disk stores sample application programs using Agilent VEE and VXI*plug&play* drivers for the 4155/4156 and the E5250A. The sample programs can control the 4155/4156, Agilent E5250A low leakage switch mainframe, and the Summit series semi-auto prober from Cascade Microtech, Inc.

The following files are stored on the disk.

- readme.txt

  This is a text file with a brief introduction of the sample programs, installation information, and so on.

- sample1.vee and sample2.vee

  These are program files that are executable on Agilent VEE (HP VEE version 4.0 or later). Refer to “What are Sample Programs?”.

- sample.ppd

  This file is an example of data used to control the Summit series semi-auto prober from Cascade Microtech, Inc. The *.ppd files will be created and used by the prober control software (PCS) furnished with the prober or supplied from Cascade Microtech, Inc. The sample application programs require the *.ppd file and PCS to control the prober.

## What are Sample Programs?

The Sample Program Disk stores two program files, sample1.vee and sample2.vee. Both programs control the 4155/4156, E5250A low leakage switch mainframe, and Cascade Summit series semi-auto prober, and do the following:

1. Probe two MOSFETs on a die
2. Measure Id-Vg characteristics of two MOSFETs
3. Extract the threshold voltage (Vth value) for two MOSFETs
4. Store the measured data into files, and display the results

The differences between the two programs are the probing control and the display, as shown in Table 7-1.

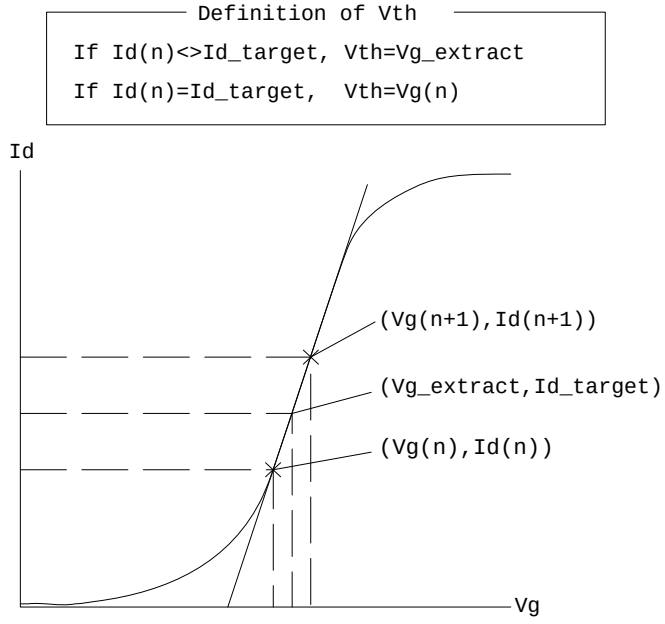
**Table 7-1 Differences Between sample1.vee and sample2.vee**

| | **sample1.vee** | **sample2.vee** |
|---|---|---|
| Probing Control | Step-by-step Measurement.<br>Probes the die where you specified the die position (x-y index) defined in the *.ppd file. | Sequential Measurement.<br>Probes sequentially for the dies defined in the *.ppd file. |
| Display | Vth value,<br>Vth pass/fail status on Wafer Map,<br>X-Y (Vg-Id) Graph of device 1,<br>X-Y (Vg-Id) Graph of device 2 | Vth value,<br>Vth pass/fail status on Wafer Map,<br>Histogram of Vth value,<br>X-Y (Vg-Id) Graph of the specified device |

## Definition of Vth

The threshold voltage (Vth) is extracted by a linear interpolation using two measurement points, which are the nearest from the targeted point for both higher and lower directions. The targeted point (Vg_extract,Id_target) is an ideal point, which indicates the Id_target value on the line through the two measurement points on the Id-Vg curve. See Figure 7-1.

**Figure 7-1** **Definition of Vth**



## Execution Mode

The sample1.vee and sample2.vee programs have five execution modes, as described below. The default is Offline mode.

**NOTE** If you do not use the Cascade Summit series semi-auto prober, use sample1.vee with Online mode, standalone, or with the E5250A. The sample2.vee program is used for sequential test, using the semi-auto prober. The test results for sample2.vee will be meaningless in the Online mode without the prober.

- Offline mode

  Select this mode if you do not use an instrument. After program execution, the demo (dummy) data is returned as the test result.

- Online mode, standalone (4155/4156 only)

  Select this mode if you use the 4155/4156 only. The test device is a single MOSFET, as the 4155/4156 has four SMUs to connect and measure a 4-terminal device simultaneously. A test fixture or manual prober is required to connect the device.

- Online mode with E5250A

  Select this mode if you use the 4155/4156 and E5250A. A test fixture or manual prober is required to connect the devices.

- Online mode with semi-auto prober

  Select this mode if you use the 4155/4156 and the Cascade Microtech Summit series semi-auto prober. The test device is a single MOSFET, as the 4155/4156 has four SMUs to connect and measure a 4-terminal device simultaneously.

- Online mode, fully automatic

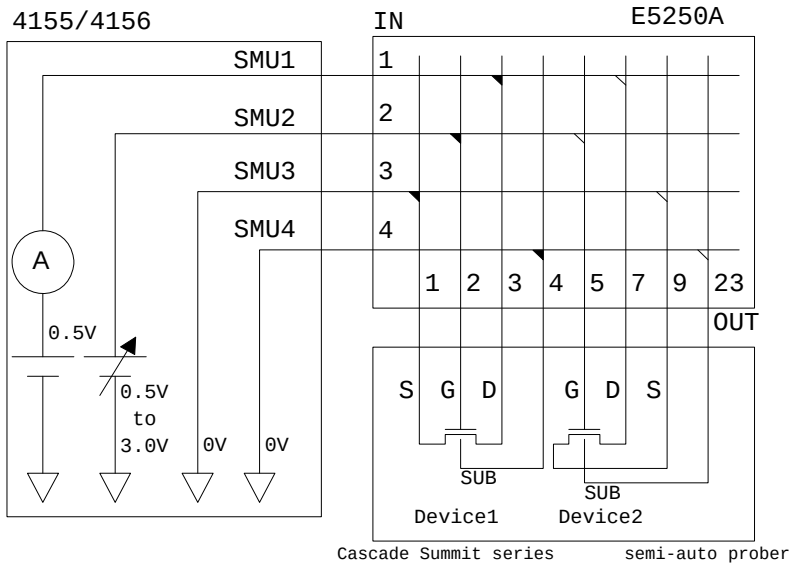
  Select this mode if you use the 4155/4156, E5250A, and the Cascade Summit series semi-auto prober.

**NOTE** The sample programs require that the E5250A be installed with two E5252A matrix cards. They must be installed in slots 1 and 2 of the E5250A mainframe.

## Measurement Connection and Source Setup

Figure 7-2 shows the measurement connection for instruments, the prober and devices. This setup is for the fully automatic Online mode. For other modes, ignore the equipment not used. This figure also shows the default source setup.

**Figure 7-2** **Measurement Connection and Source Setup**




**NOTE** To avoid misconnection, pay close attention to the die (device1 and device2) and the pin assignment of the probe card, if used.

**NOTE** In Figure 7-2, OUT 1 to 9 indicates output port 1 to 9 of the matrix card installed in slot 1 of the E5250A; OUT 23 indicates output port 11 of the card installed in slot 2 of the E5250A.

## Measurement Data Files

The sample programs create three types of data files, as shown below.

| | |
|---|---|
| info.txt | Information file. Title, date, and comment are included. |
| vth*n*.txt | Vth data file. If 5 dies are tested, 5 data are included. For the file name, *n* is 1 or 2 (1 for device1, 2 for device2). |
| d*n*vgid*xy*.txt | Row measurement data file of Id-Vg curve. If 5 dies are tested, the program creates 10 data files (5 files/device). The file names *n*, *x*, and *y* indicate the following:<br>*n*: 1 or 2 (1 for device1, 2 for device2).<br>*x*: X-index of the die position.<br>*y*: Y-index of the die position. |

**Figure 7-3 Example of Data Files Created by Sample Programs**

info.txt

```
Data Save Directory for HP4155B/4156B Sample Program
Mon 15/Jun/1998 14:14:30
Comment :
```

vth1.txt

```
Device 1 Vth Table
X Index        Y Index        "Vth [V]"
5              7              9.21E-01
7              5              9.19E-01
5              5              7.32E-01
3              5              9.20E-01
5              3              9.22E-01
```

vth2.txt

```
Device 2 Vth Table
X Index        Y Index        "Vth [V]"
5              7              9.10E-01
7              5              9.19E-01
5              5              9.06E-01
3              5              9.22E-01
5              3              9.09E-01
```

d1vgid55.txt

```
(5,5) Device 1 Vg-Id Data
Vg [V]         Id [A]         Id Status
-5.00E-01      8.00E-15       0
-4.13E-01      8.00E-15       0
-3.25E-01      7.06E-15       0
-2.38E-01      6.06E-15       0
-1.50E-01      6.00E-15       0
-6.25E-02      6.00E-15       0
2.50E-02       5.06E-15       0
1.13E-01       6.89E-15       0
2.00E-01       2.87E-14       0
2.88E-01       4.61E-13       0
3.75E-01       9.17E-12       0
4.63E-01       1.66E-10       0
5.50E-01       2.91E-09       0
```

d2vgid55.txt

```
(5,5) Device 2 Vg-Id Data
Vg [V]         Id [A]         Id Status
-5.00E-01      9.00E-15       0
-4.13E-01      8.00E-15       0
-3.25E-01      8.00E-15       0
-2.38E-01      8.00E-15       0
-1.50E-01      7.00E-15       0
-6.25E-02      6.00E-15       0
2.50E-02       6.00E-15       0
1.13E-01       6.00E-15       0
2.00E-01       5.00E-15       0
2.88E-01       6.99E-15       0
3.75E-01       2.99E-14       0
4.63E-01       4.85E-13       0
5.50E-01       9.67E-12       0
```

# Installation

This section explains the equipment and accessories required to use the sample programs, and how to install the programs.

## Required Equipment and Accessories

1. PC (AT-compatible) for Windows

   Agilent VEE (HP VEE version 4.0 or later) and the VXI*plug&play* drivers for the 4155/4156 and E5250A must be installed in your PC and ready for use. See Chapter 1.

   A 3.5 inch flexible disk drive must be connected to your PC to install the sample programs.

   If you use the Cascade Microtech Summit series semi-auto prober, the prober control software (PCS) supplied by Cascade Microtech, Inc. must be installed in your PC and the prober must be connected to your PC via Cascade's interface. For the interface, and the operating system supported by the PCS, contact Cascade Microtech, Inc.

2. Agilent VEE Sample Program Disk
3. Agilent 4155 or 4156 Semiconductor Parameter Analyzer
4. Agilent E5250A Low Leakage Switch Mainframe

   Two E5252A matrix cards are required and must be installed in slot 1 and 2 of the E5250A.

5. Cascade Microtech Summit Series Semi-Auto Prober

   If you do not use the semi-auto prober, prepare the manual prober or test fixture, such as Agilent 16442A/B, to connect the test devices.

6. Connection Cables

   Four triaxial cables are required to connect the 4155/4156 and E5250A.

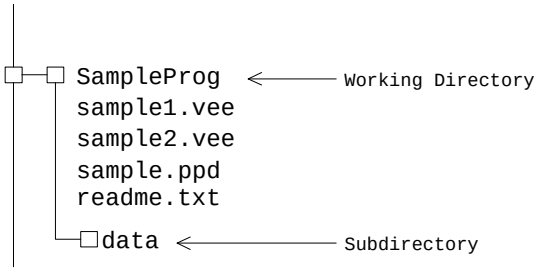
   Eight triaxial cables are required to connect the E5250A and the prober or test fixture. You will also need coaxial cables, probe card, manipulators, or wire to connect the test devices.

   Two GPIB cables are required to connect the instruments and the PC.

## Installing the Sample Programs

1. Make a working directory where you can install and execute the sample programs, using Windows Explorer.
2. Create a subdirectory in the working directory. The subdirectory will be used to save the measurement data files.
3. Insert the Agilent VEE Sample Program Disk into the flexible disk drive connected to your PC.
4. Copy all of the files on the disk to the working directory.

**Figure 7-4** **Installing Sample Programs**



# Using sample1.vee

This section covers the following topics.

- “Program Execution Flow”
- “Panel Display”
- “To Execute sample1.vee”

**NOTE**

For the wafer test using the Summit series semi-auto prober from Cascade Microtech, Inc., create your own probe plan file (*.ppd). The sample.ppd file stored on the Agilent VEE Sample Program Disk is an example only.

## Program Execution Flow

The execution flow of the sample1.vee program is shown in Figure 7-5 and Table 7-2.

**Figure 7-5 Execution Flow of sample1.vee**

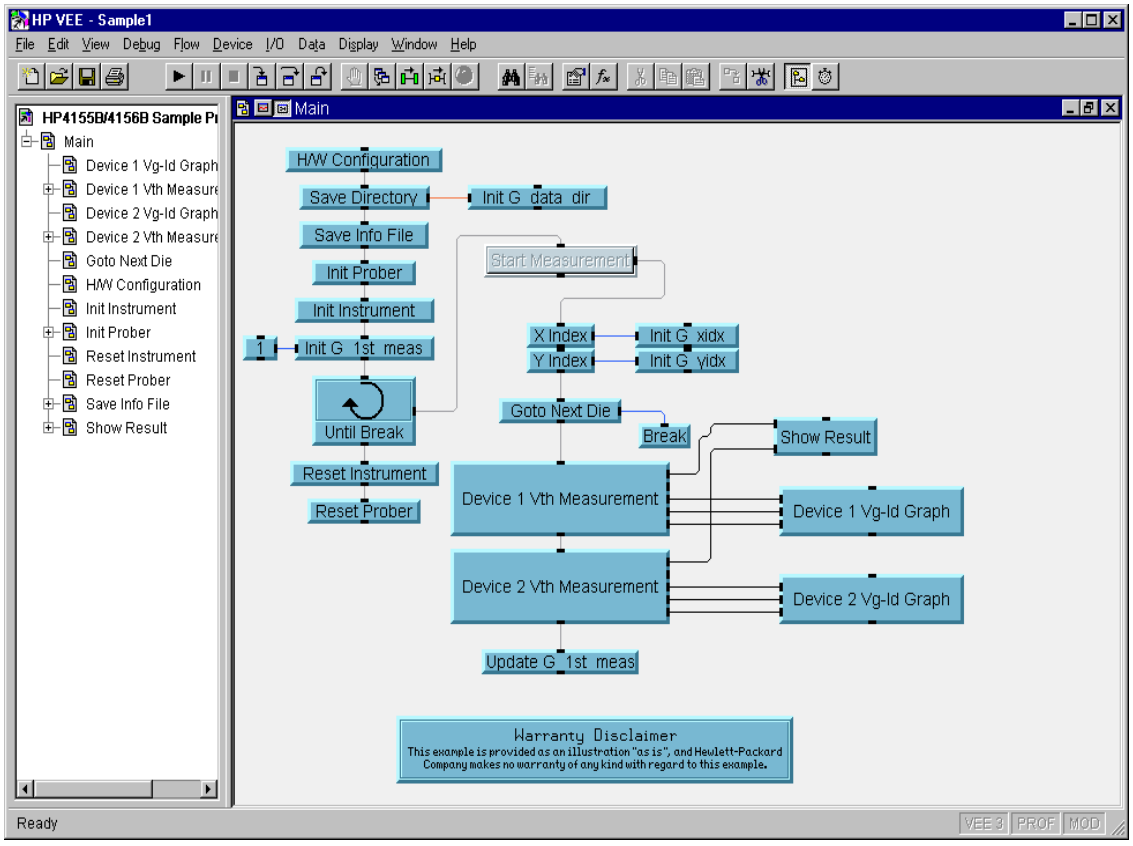


**Table 7-2 Execution Flow of sample1.vee**

| | Object | Explanation |
|---|---|---|
| 1 | H/W Configuration | Defines execution mode. Set the mode before running sample1.vee. |
| 2 | Save Directory,<br>Init G data dir | Defines name of the subdirectory to save the measurement result data. See “Installing the Sample Programs” on page 10. |
| 3 | Save Info File | Defines and saves the information file (info.txt) which contains title, date, and comments for the subdirectory. You can enter comments. See “Measurement Data Files” on page 8. |
| 4 | Init Prober | Initializes Cascade Summit series semi-auto prober, if used. |
| 5 | Init Instrument | Initializes instrument, if used. |
| 6 | 1, Init G 1st meas | Sets G_1st_meas value. If G_1st_meas=1, header lines are written in vth1.txt and vth2.txt. See “Measurement Data Files” on page 8. |
| 7 | Until Break | Repeats the following sequence until a break occurs. |
| 8 | Start Measurement | Triggers the start of the measurement. |
| 9 | X Index, Init G xidx,<br>Y Index, Init G yidx | Defines X and Y index of the die tested. The index must be defined in the *.ppd file used. |
| 10 | Goto Next Die, Break | Probes the die specified by the X-Y index. Breaks if X index < 0. |
| 11 | Device1 Vth Measurement<br>Device2 Vth Measurement | Executes Id-Vg measurement, extracts Vth, and saves measurement results. See “Measurement Data Files” on page 8. |
| 12 | Device1 Vg-Id Graph<br>Device2 Vg-Id Graph | Displays Id-Vg measurement result graphs of Device 1 and 2. See Figure 7-6. |
| 13 | Show Result | Displays Vth value and wafer map. See Figure 7-6. Dev1 Vth and Dev2 Vth show Vth value, and the field below shows wafer map. |
| 14 | Update G 1st meas | Sets G_1st_meas variable to 0. |
| 15 | Reset Instrument | Resets the instruments. |
| 16 | Reset Prober | Resets the prober. |

## Panel Display

The sample1.vee program displays the following data and graph. See Figure 7-6.

| | |
|---|---|
| Vth | Displays Vth value of device 1 and 2. The data is in volts. |
| Vg-Id Graph | Displays Id-Vg curve of device 1 and 2. |
| X, Y Index | Displays wafer map of Vth value using the following characters. |

.: Both device 1 and 2 test data are within the allowable range.

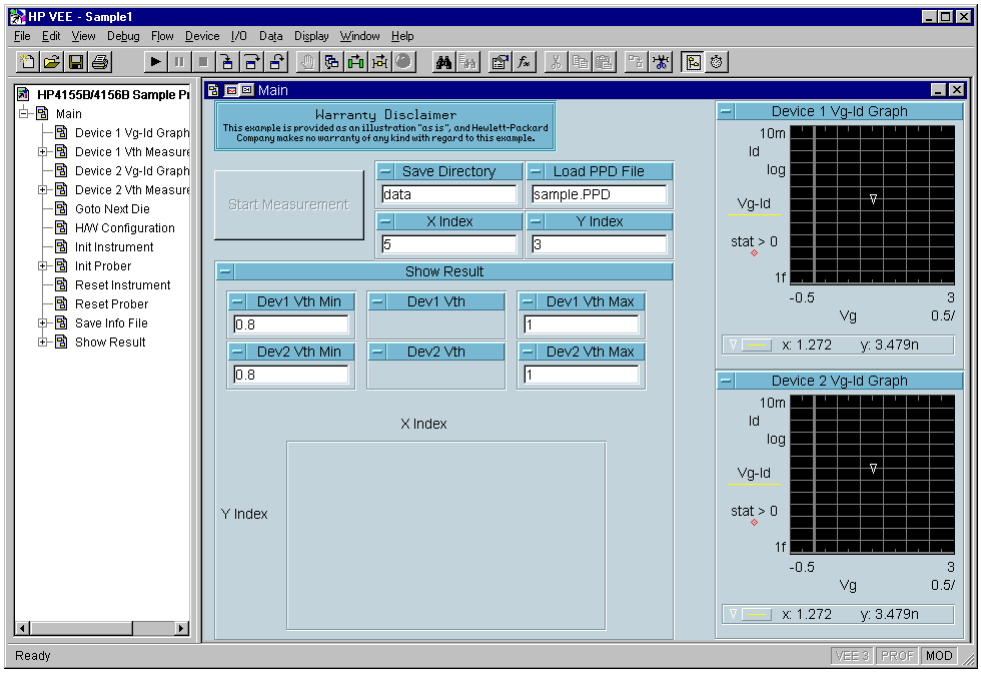
F1: Device 1 test data is out of the allowable range.

F2: Device 2 test data is out of the allowable range.

F3: Both device 1 and 2 test data are out of the allowable range.

The allowable range is specified by Dev1(2) Vth Min and Dev1(2) Vth Max input fields. Min field sets the lower limit, and Max field sets the upper limit.

**Figure 7-6 Panel Display of sample1.vee**



## To Execute sample1.vee

Before executing the sample1.vee program, do the following.

**NOTE** If you execute sample1.vee in Offline mode, skip steps 1 through 5.

1. Connect GPIB cables between your PC and the instruments being used.
2. Confirm that the semi-auto prober is connected to your PC via Cascade's interface, or connect the prober to your PC, if used.
3. Connect the measurement cables between the instruments and the prober or test fixture being used. See "Measurement Connection and Source Setup" on page 7.
4. Turn on the instruments and the semi-auto prober being used, if applicable.
5. Display the SYSTEM: MISCELLANEOUS screen on the 4155/4156. Select NOT SYSTEM CONTROLLER in the *4155C/4156C is* field.
6. Run Agilent VEE. If this is the first time using Agilent VEE and VXI*plug&play* drivers for the 4155/4156 and E5250A, register the drivers at this time. See "Programming Basics" on page6-3.
7. Open the sample1.vee program.
8. Display the program (Figure 7-5) and double click the H/W Configuration object. The panel of this object is displayed.
9. On the panel, select the check button of the instruments being used and the semi-auto prober, if used. See Table 7-3.

**Table 7-3 H/W Configuration Object Check Button Setup**

| Execution Mode | 4155/56 | E5250A | Semi-Auto Prober |
|---|---|---|---|
| Online, standalone | check | | |
| Online, with E5250A | check | check | |
| Online, with prober | check | | check |
| Online, fully automatic | check | check | check |
| Offline | | | |

To execute the sample1.vee program, do the following.

**NOTE** If you execute sample1.vee in Offline mode, skip steps 3, 5, and 6.

1. Create a directory (Example: C:\lot1\test1\data) to be used to save the measurement data. To create a directory, use Windows Explorer. See “Installing the Sample Programs” on page 10.
2. Display the panel (Figure 7-6) and enter the following input fields.

   | | |
   |---|---|
   | Save Directory | Enter the name of the directory to save measurement data. Enter only the name if the directory is under the current directory which this program is stored, or enter the entire path to specify another directory, such as C:\lot1\test1\data. |
   | Load PPD File | Enter the file name of the probe plan data file (*.ppd) for the Cascade Microtech Prober Control Software. Ignore this field if you do not use the semi-auto prober. |
   | Dev1 Vth Min/Max | Enter the allowable range of device 1 Vth value. Min field sets the lower limit, Max field sets the upper limit. |
   | Dev2 Vth Min/Max | Enter the allowable range of device 2 Vth value. Min field sets the lower limit, Max field sets the upper limit. |

3. Connect the device.

   If you use the semi-auto prober, load a wafer on the prober, and keep the platen handle up.

   If you do not use the semi-auto prober, connect devices (two MOSFETs) to a test fixture, or load a wafer on a manual prober and probe a die tested.

4. Click the run button on the Agilent VEE menu bar. If you use the semi-auto prober, the Wait the DDE Server setup dialog box is displayed, and the Cascade Microtech prober control software is called. See Figure 7-7

   If you do not use the semi-auto prober, skip steps 5 and 6.

5. Click Continue. A window for the prober control software is displayed as shown in Figure 7-8.

   This example shows the SAMPLE.PPD window of the prober control software. The title of the window will be the file name you entered in the Load PPD File input field in step 2.

6. Move the wafer to align the probes over the probe plan alignment position, then click OK in the ALIGN PROBES dialog box.

**Figure 7-7** **Running sample1.vee with Cascade Microtech Prober Control Software (1)**

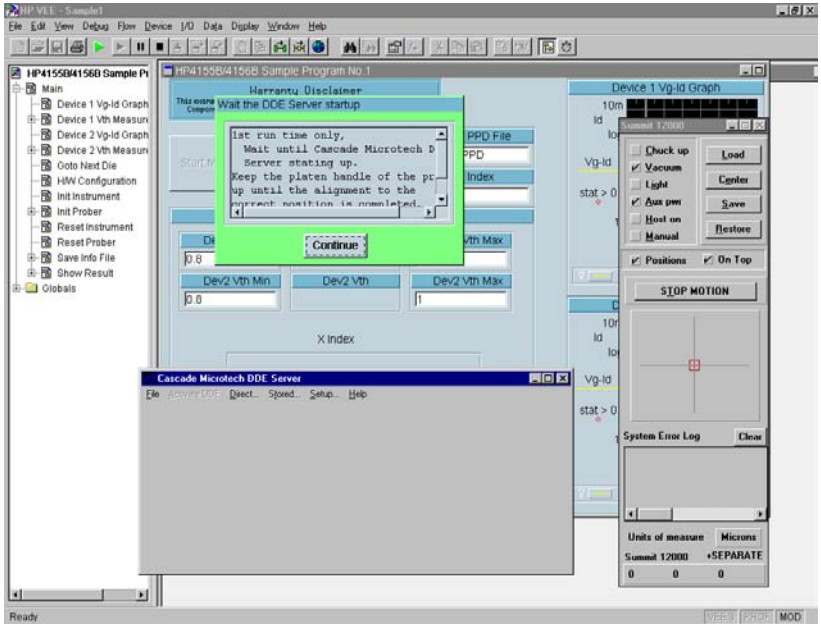
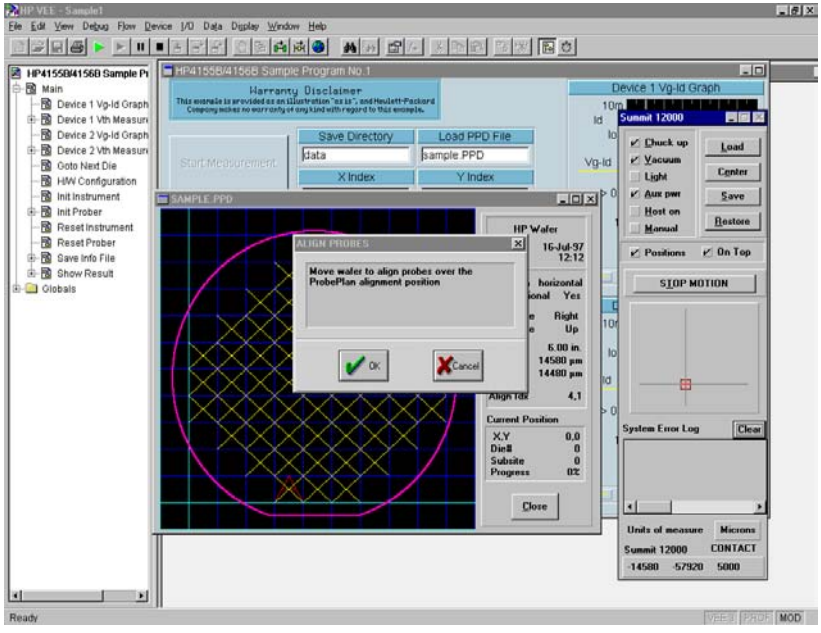
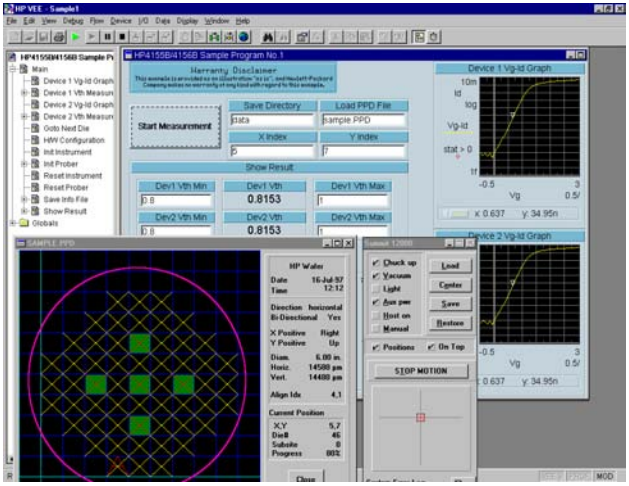



**Figure 7-8** **Running sample1.vee with Cascade Microtech Prober Control Software (2)**



7. Enter the X-Y index of the die to be tested in the X Index and Y Index input fields. Only the index defined in the *ppd file is effective for this test.
8. Click Start Measurement. The program executes the Id-Vg measurement, extracts Vth value, displays the results, and stores the data into files. The program then waits for your input.

**Figure 7-9** **Execution Example of sample1.vee**




9. Repeat steps 7 and 8 for all dies to be tested.
10. To stop the program, click the stop button on the Agilent VEE menu bar.

**NOTE** In Offline mode, the program returns the dummy data instead of the raw measurement data in step 8.

**NOTE** A wafer map is also displayed in the Cascade Microtech Prober Control Software *.ppd window. This window indicates results by using the following color scheme.

| | |
|---|---|
| Green: | Both device 1 and 2 test data are within the allowable range. |
| Yellow: | Device 1 test data is out of the allowable range. |
| Magenta: | Device 2 test data is out of the allowable range. |
| Red: | Both device 1 and 2 test data are out of the allowable range. |

**NOTE** To exit the Cascade Microtech prober control software, select the File-Exit menu of the Cascade Microtech DDE Server window, then click Yes in the Halt Cascade DDE Server dialog box.

# Using sample2.vee

This section covers the following topics.

- “Program Execution Flow”
- “Panel Display”
- “To Execute sample2.vee”

**NOTE**

For the wafer test using the Summit series semi-auto prober from Cascade Microtech, Inc., create your probe plan file (*.ppd). The sample.ppd file stored on the Agilent VEE Sample Program Disk is an example only.

## Program Execution Flow

The execution flow of the sample2.vee program is shown in Figure 7-10 and Table 7-4.

**Figure 7-10** **Execution Flow of sample2.vee**

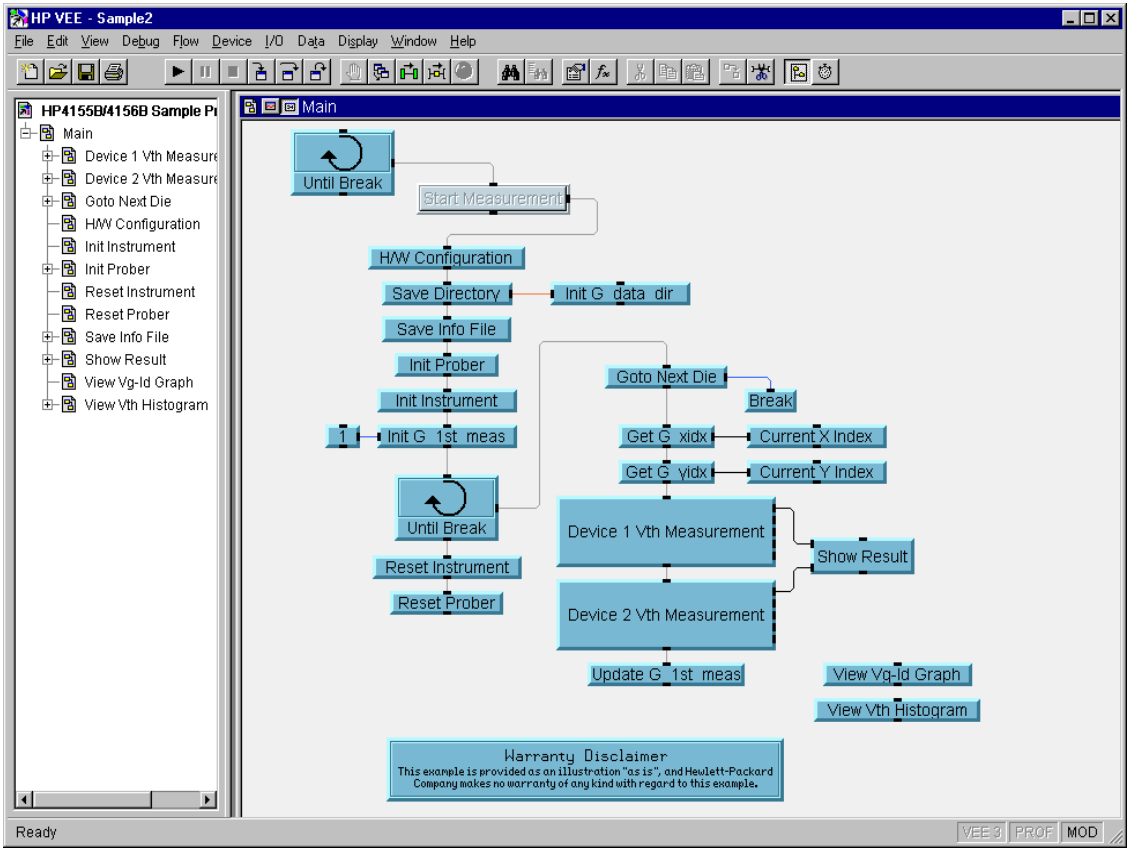


**Table 7-4 Execution Flow of sample2.vee**

| | Object | Explanation |
|---|---|---|
| 1 | Until Break | Repeats the following sequence until a break occurs. |
| 2 | Start Measurement | Triggers the start of the wafer test. |
| 3 | H/W Configuration | Defines execution mode. Set the mode before running sample2.vee. |
| 4 | Save Directory,<br>Init G data dir | Defines name of the subdirectory to save measurement result data. See “Installing the Sample Programs” on page 10. |
| 6 | Save Info File | Defines and saves the information file (info.txt) which contains title, date, and comments for the subdirectory. You can enter comments. See “Measurement Data Files” on page 8. |
| 7 | Init Prober | Initializes Cascade Summit series semi-auto prober, if used. |
| 8 | Init Instrument | Initializes instruments, if used. |
| 9 | 1, Init G 1st meas | Sets G_1st_meas value. If G_1st_meas=1, the prober sets the first die to probe, and header lines are written in vth1.txt and vth2.txt. See “Measurement Data Files” on page 8. |
| 10 | Goto Next Die, Break | Probes the die to test. The die and probing sequence depend on the *.ppd file used. Breaks if the test was completed for all die. |
| 11 | Get G xidx, Get G yidx<br>Current X, Y Index | Gets and displays the X-Y index of the die now tested. |
| 13 | Device1 Vth Measurement<br>Device2 Vth Measurement | Executes Id-Vg measurement, extracts Vth, and saves measurement results. See “Measurement Data Files” on page 8. |
| 14 | Show Result | Displays Vth value and wafer map. See Figure 7-11. Dev1 Vth and Dev2 Vth show Vth value, and the field below shows wafer map. |
| 15 | Update G 1st meas | Sets G_1st_meas variable to 0. |
| 16 | Reset Instrument | Resets the instruments. |
| 17 | Reset Prober | Resets the prober. |
| -- | View Vg-Id Graph | Displays Id-Vg curve of the device you select. |
| -- | View Vth Histogram | Displays histogram for Vth of the device you select (device 1 or device 2). |

## Panel Display

The sample2.vee program displays the following data and graph. See Figure 7-11.

| | |
|---|---|
| Vth | Displays Vth value of device 1 and 2. The data is in volts. |
| Histogram | Displays histogram of Vth value for the device selected. Enter the file name, then click View to display the histogram. |
| Vg-Id Graph | Displays Id-Vg curve for the device selected. Enter the file name, then click View to display the graph. |
| X, Y Index | Displays wafer map of Vth value using the following characters. |

. : Both device 1 and 2 test data are within the allowable range.

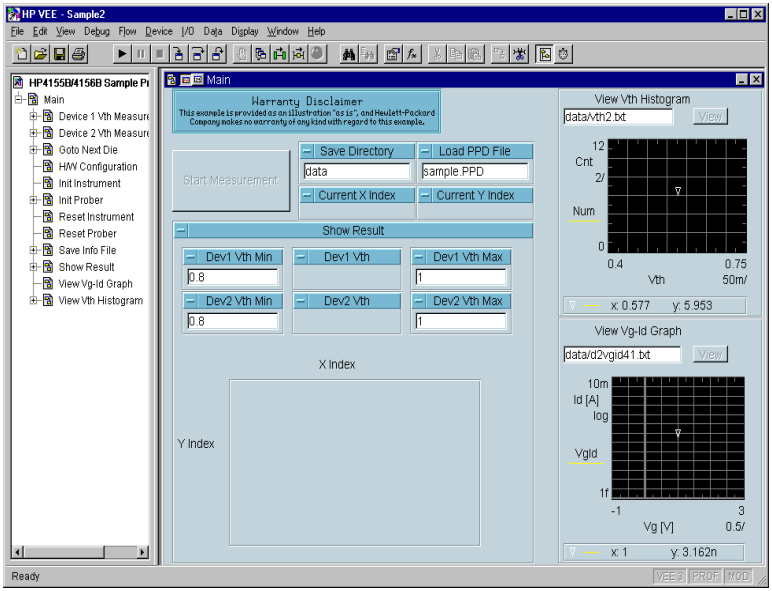
F1: Device 1 test data is out of the allowable range.

F2: Device 2 test data is out of the allowable range.

F3: Both device 1 and 2 test data are out of the allowable range.

The allowable range is specified by Dev1(2) Vth Min and Dev1(2) Vth Max input fields. Min field sets the lower limit, and Max field sets the upper limit.

**Figure 7-11** **Panel Display of sample2.vee**



## To Execute sample2.vee

Before executing the sample2.vee program, do the following.

**NOTE** If you execute sample2.vee in Offline mode, skip steps 1 through 5.

1. Connect the GPIB cables between your PC and the instruments being used.
2. Confirm that the semi-auto prober is connected to your PC via Cascade's interface, or connect the prober to your PC, if used.
3. Connect the measurement cables between the instruments and the prober or test fixture used. See "Measurement Connection and Source Setup" on page 7.
4. Turn on the instruments and the semi-auto prober being used, if applicable.
5. Display the SYSTEM: MISCELLANEOUS screen on the 4155/4156. Then select NOT SYSTEM CONTROLLER in the *4155C/4156C is* field.
6. Run Agilent VEE. If this is the first time using Agilent VEE and VXI*plug&play* drivers for the 4155/4156 and E5250A, register the drivers at this time. See "Programming Basics" on page6-3.
7. Open the sample2.vee program.
8. Display the program (Figure 7-10) and double click the H/W Configuration object. The panel for the object is displayed.
9. On the panel, select the check button of the instruments and the semi-auto prober being used, if applicable. See Table 7-5.

**Table 7-5 H/W Configuration Object Check Button Setup**

| Execution Mode | 4155/56 | E5250A | Semi-Auto Prober |
|---|---|---|---|
| Online, standalone | check | | |
| Online, with E5250A | check | check | |
| Online, with prober | check | | check |
| Online, fully automatic | check | check | check |
| Offline | | | |

To execute the sample2.vee program, do the following.

**NOTE** If you execute sample2.vee in Offline mode, skip steps 4, 6, and 7.

1. Click the run button on the Agilent VEE menu bar.
2. Create a directory (Example: C:\lot1\test1\data) to save the measurement data. To create a directory, use Windows Explorer. See “Installing the Sample Programs” on page 10.
3. Enter the following input fields.

   | | |
   |---|---|
   | Save Directory | Enter the name of the directory to save measurement data. Enter only the name if the directory is under the current directory where this program is stored, or enter the entire path to specify a different directory, such as C:\lot1\test1\data. |
   | Load PPD File | Enter the file name of the probe plan data (*.ppd) file for Cascade Microtech Prober Control Software. Ignore this field if the semi-auto prober is not used. |
   | Dev1 Vth Min/Max | Enter the allowable range of Vth value for device 1. Min field sets the lower limit, Max field sets the upper limit. |
   | Dev2 Vth Min/Max | Enter the allowable range of Vth value for device 2. Min field sets the lower limit, Max field sets the upper limit. |

4. Connect the device.

   If you use the semi-auto prober, load a wafer on the prober, and keep the platen handle up.

   If you do not use the semi-auto prober, connect devices (two MOSFETs) to a test fixture, or load a wafer on a manual prober and probe a die tested.

5. Click Start Measurement. If you use the semi-auto prober, the Wait the DDE Server setup dialog box is displayed, and the Cascade Microtech prober control software is called. See Figure 7-12

   If you do not use the semi-auto prober, skip steps 6 and 7.

6. Click Continue. A window of the prober control software is displayed as shown in Figure 7-13.

   This example shows the SAMPLE.PPD window of the prober control software. The title of the window will be the file name you entered in the Load PPD File input field in step 3.

**Figure 7-12** **Running sample2.vee with Cascade Microtech Prober Control Software (1)**

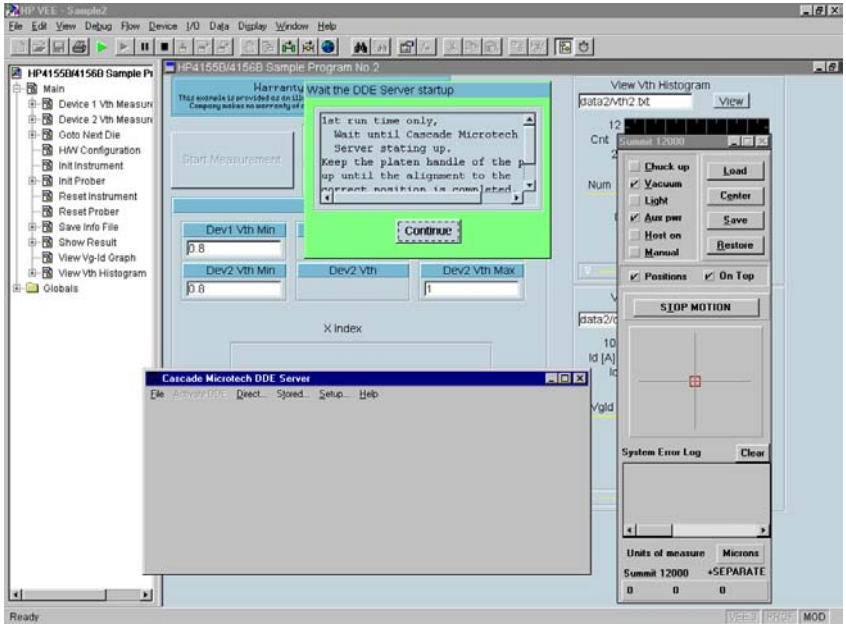
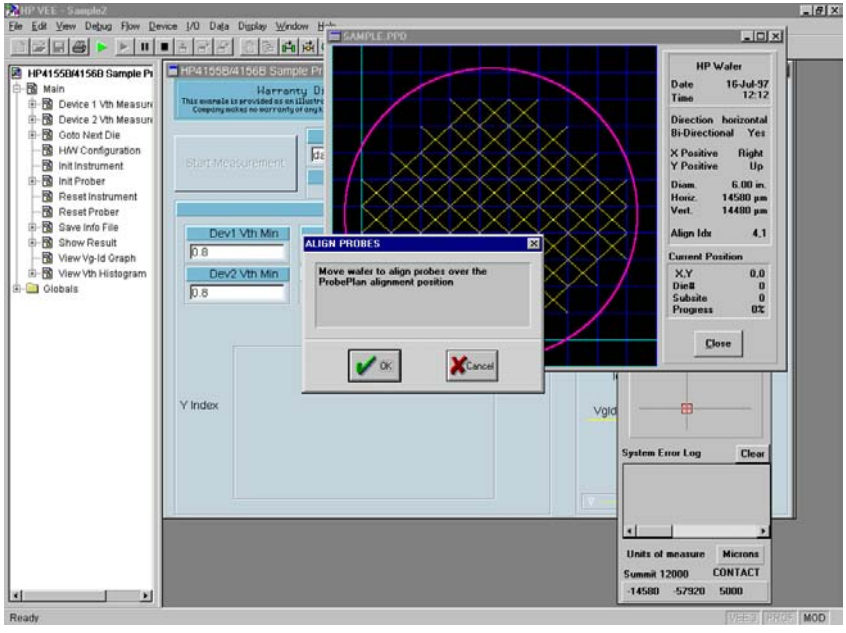
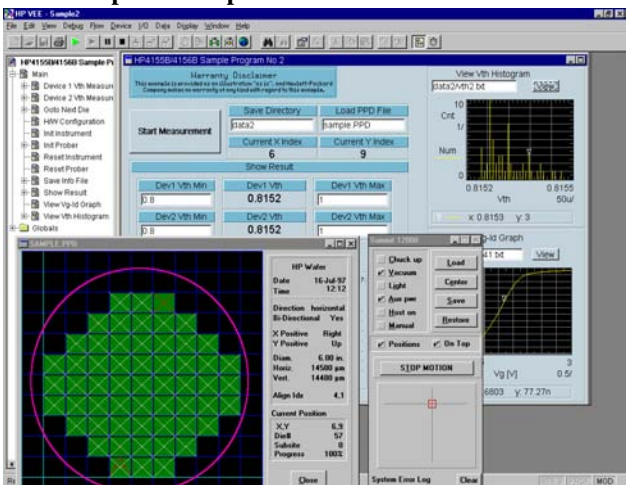



**Figure 7-13** **Running sample2.vee with Cascade Microtech Prober Control Software (2)**



7. Move the wafer to align the probes over the probe plan alignment position, then click OK in the ALIGN PROBES dialog box.
8. Wait until wafer test is completed. The program executes the Id-Vg measurement, extracts Vth value, displays the results, and stores the data into files. The program then waits for your input.

**Figure 7-14** **Execution Example of sample2.vee**



9. Repeat step 2 through 8 for all wafers to be tested.
10. To stop the program, click the stop button on the Agilent VEE menu bar.

**NOTE** In Offline mode, the program returns the dummy data instead of the raw measurement data in step 8.

**NOTE** A wafer map is also displayed in the Cascade Microtech Prober Control Software *.ppd window. The window indicates the results by using the following color scheme.

| | |
|---|---|
| Green: | Both device 1 and 2 test data are within the allowable range. |
| Yellow: | Device 1 test data is out of the allowable range. |
| Magenta: | Device 2 test data is out of the allowable range. |
| Red: | Both device 1 and 2 test data are out of the allowable range. |

**NOTE** To exit the Cascade Microtech Prober Control Software, select the File-Exit menu of the Cascade Microtech DDE Server window. Then click Yes in the Halt Cascade DDE Server dialog box.

# Customizing Sample Programs

This section offers examples of modifications to the sample1.vee and sample2.vee programs:

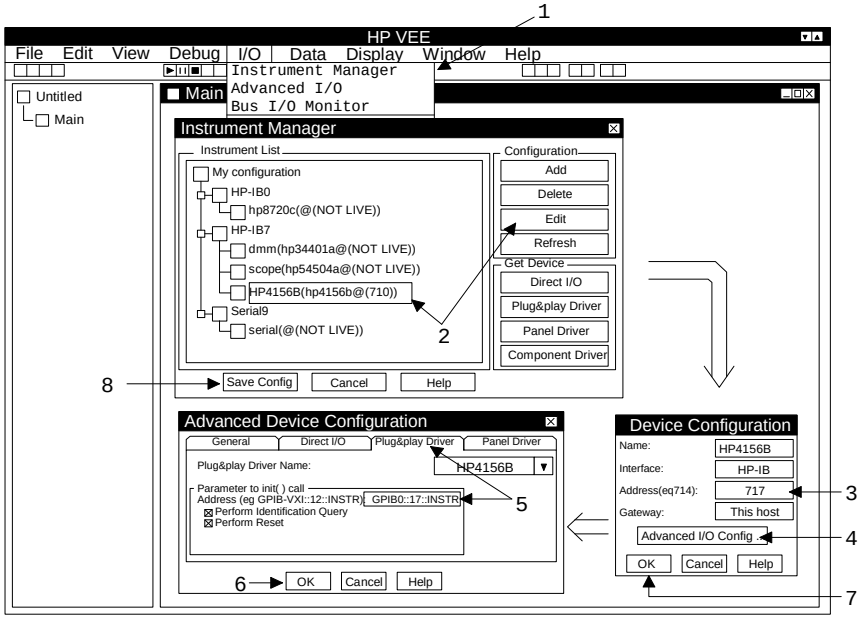
- “To Change an GPIB Address”
- “To Change the Vth Measurement Setup”
- “To Remove a Test Device”
- “To Remove a Source Output”
- “To Add a Test Device”
- “To Add a Measurement Parameter”

# To Change an GPIB Address

You can change the GPIB address of the 4155/4156 and E5250A by using the Agilent VEE Instrument Manager.

1. Select the I/O-InstrumentManager menu from the Agilent VEE menu bar. The Instrument Manager dialog box is displayed.
2. Select the instrument (Example: HP4156B) from the Instrument Manager dialog box, and then click Edit. The Device Configuration dialog box is displayed.
3. Enter the new address in the Address field. For example, enter “717”.
4. Click Advanced I/O Config. The Advanced Device Configuration dialog box is displayed.
5. Select the Plug&play Driver Tab, and then enter the new address in the Address field. For example, enter “GPIB0::17::INSTR”.
6. Click OK in the Advanced Device Configuration dialog box.
7. Click OK in the Device Configuration dialog box.
8. Click Save Config in the Instrument Manager dialog box to register the new address.

**Figure 7-15** **To Change an GPIB Address**



## To Change the Vth Measurement Setup

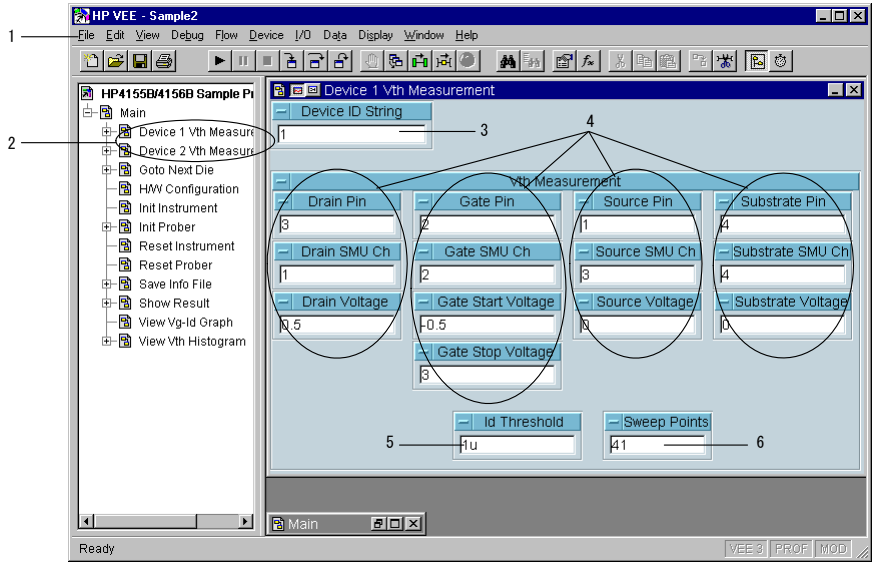
You can change the 4155/4156 source setup and the E5250A switching setup by editing the Device 1 (or 2) Vth Measurement object. See Figure 7-16.

1. Open the sample1.vee or sample2.vee program.
2. Double click Device 1 (or 2) Vth Measurement in the Agilent VEE program explorer. The Device 1 (or 2) Vth Measurement object is displayed.
3. Change the Device ID for MOSFET if needed, using a string format.
4. Change the setup for all terminals for MOSFET.

   | | |
   |---|---|
   | Pin | E5250A output channel number; 1 to 24 are available. |
   | SMU Ch | 4155/4156 SMU number; 1 to 4 are available. |
   | Voltage | SMU output voltage (in volts). |
   | Start,Stop Voltage | SMU output voltage for Vg sweep (in volts). |

5. Change the target Id for extracting Vth. See “Definition of Vth” on page 5.
6. Change the number of measurement points in a sweep; 2 to 256 are available.

**Figure 7-16 Vth Measurement Panel Display**



**NOTE**

If you want to change other source setup parameters, such as compliance, you will need to change the setup of the To/From object, by doing the following.

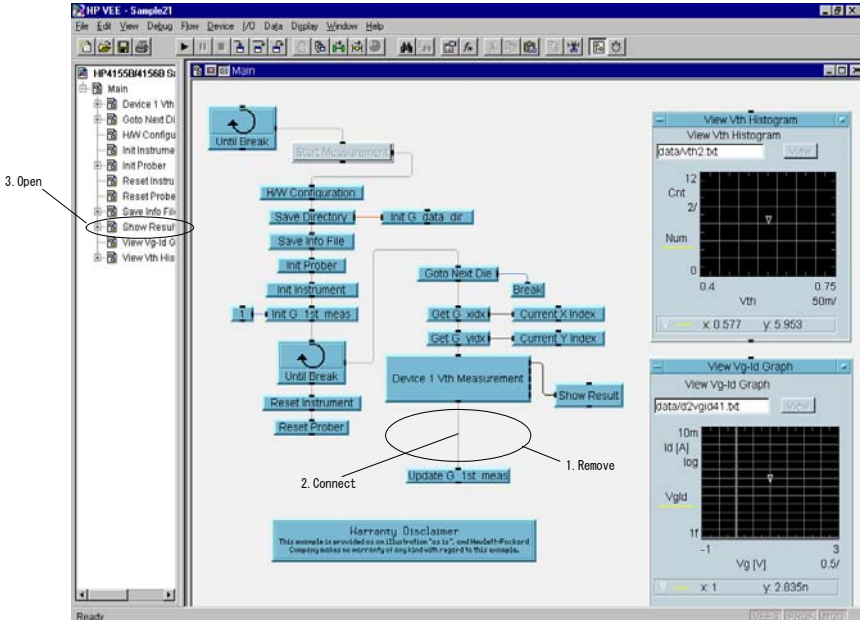
1. Display the program for the Device 1 (or 2) Vth Measurement object.
2. Open the Vth Measurement object, and display the program.
3. Open the Pre-Setup object, and then open the Setup4155 object.
4. Double click hp4156b_force and display the Edit Function Panel.
5. Change the setup value on the panel.

## To Remove a Test Device

If your test die includes only one MOSFET, modify the program as shown below. This example modifies sample2.vee, and removes objects for device 2.

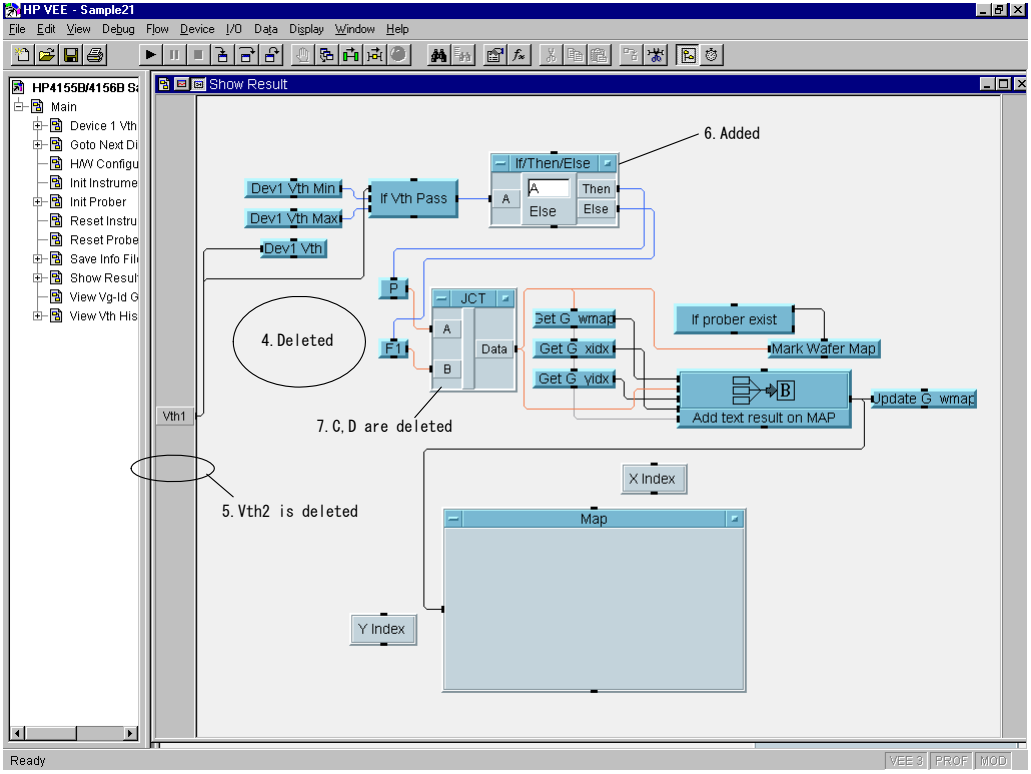
1. Cut the Device 2 Vth Measurement object from the Main program display.
2. Connect the control line between the Device 1 Vth Measurement object and the Update G 1st meas object.
3. Open the Show Result object using the Agilent VEE program explorer.

**Figure 7-17 To Remove a Test Device (1)**



4. Display the Show Result object program, and cut the following seven objects.
   - Dev2 Vth, Dev2 Vth Min, Dev2 Vth Max, If Vth Pass (for device2)
   - If/Then/Else
   - F2, F3
5. Delete the input terminal Vth2.
6. Add the Flow-If/Then/Else object, and enter A, then connect the lines:
   - between If Vth Pass and If/Then/Else A terminal
   - between If/Then/Else Then terminal and P
   - between If/Then/Else Else terminal and F1
7. Double click the JCT object, and delete the input terminals C and D.

**Figure 7-18 To Remove a Test Device (2)**



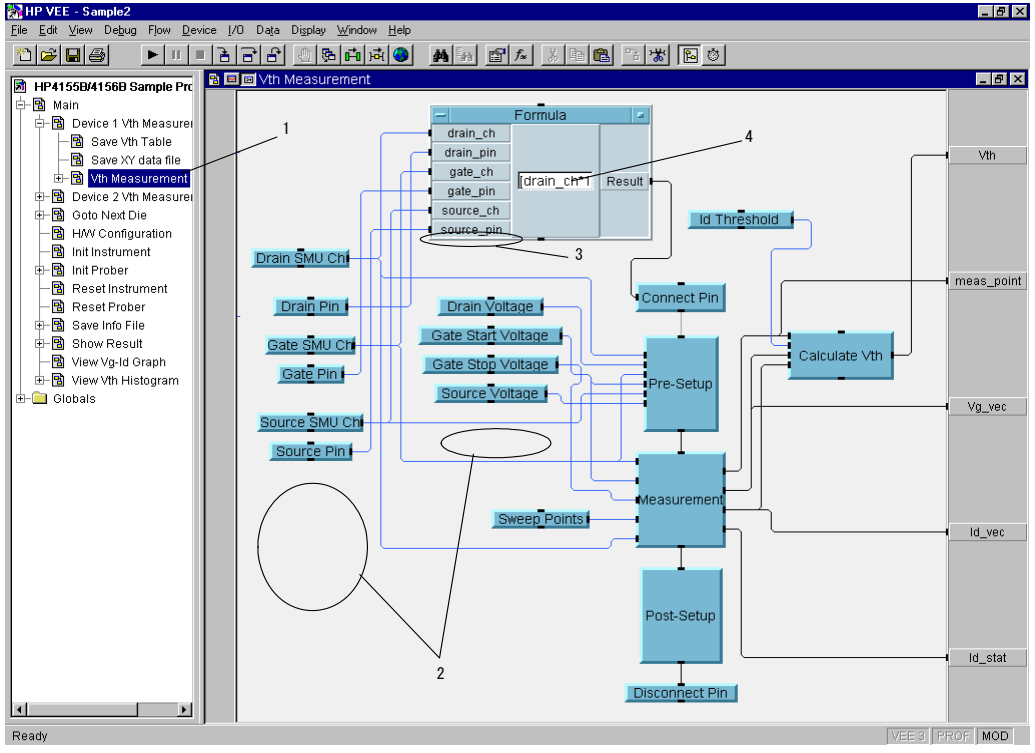
## To Remove a Source Output

If the number of terminals for your device is less than four, modify the program as shown below. This example modifies sample2.vee, and removes objects for the substrate terminal of device 1 (the example does not modify the objects for device 2).

1. Open the Vth Measurement object using the Agilent VEE program explorer.
2. Display the program, and cut the following three objects.
   - Substrate SMU Ch
   - Substrate Pin
   - Substrate Voltage
3. Open the Formula object, and delete the sub_ch and sub_pin input terminals.
4. Change the formula, defined in the Formula object, as shown below.

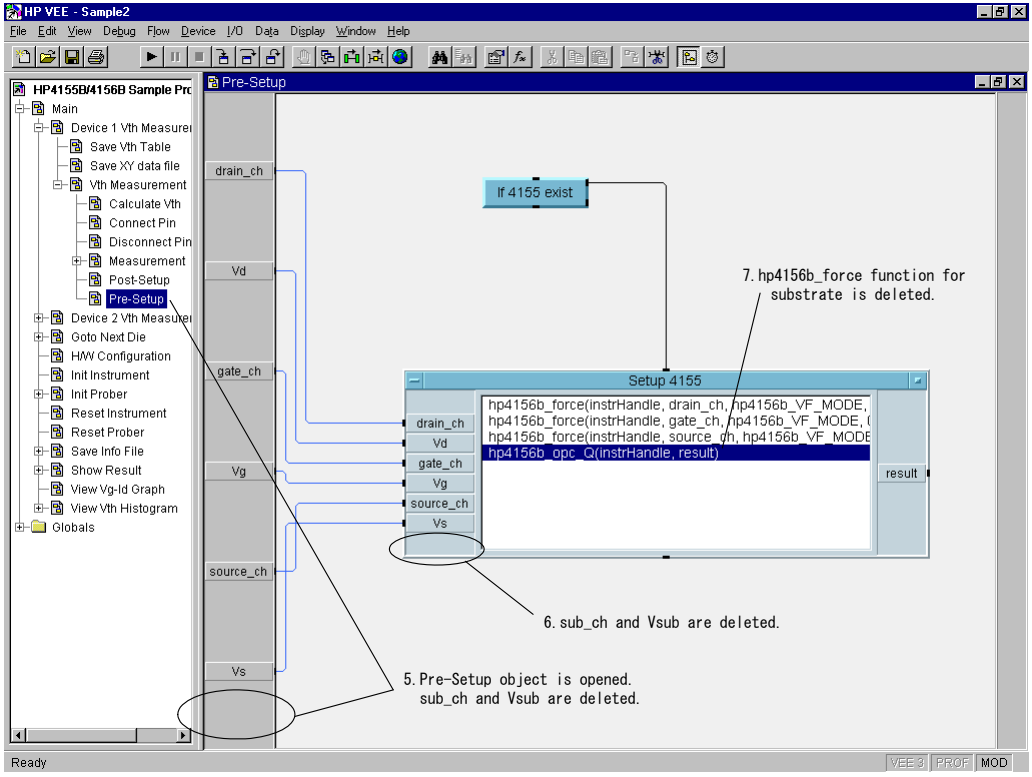
   [drain_ch*100+drain_pin, gate_ch*100+gate_pin, source_ch*100+source_pin, 0]

**Figure 7-19 To Remove a Source Output (1)**



5. Open the Pre-Setup object, and delete the sub_ch and Vsub input terminals.
6. Open the Setup 4155 object, and delete the sub_ch and Vsub input terminals.
7. Delete the hp4156b_force(instrHandle,sub_ch,.....) function from the Setup 4155 object.

**Figure 7-20 To Remove a Source Output (2)**

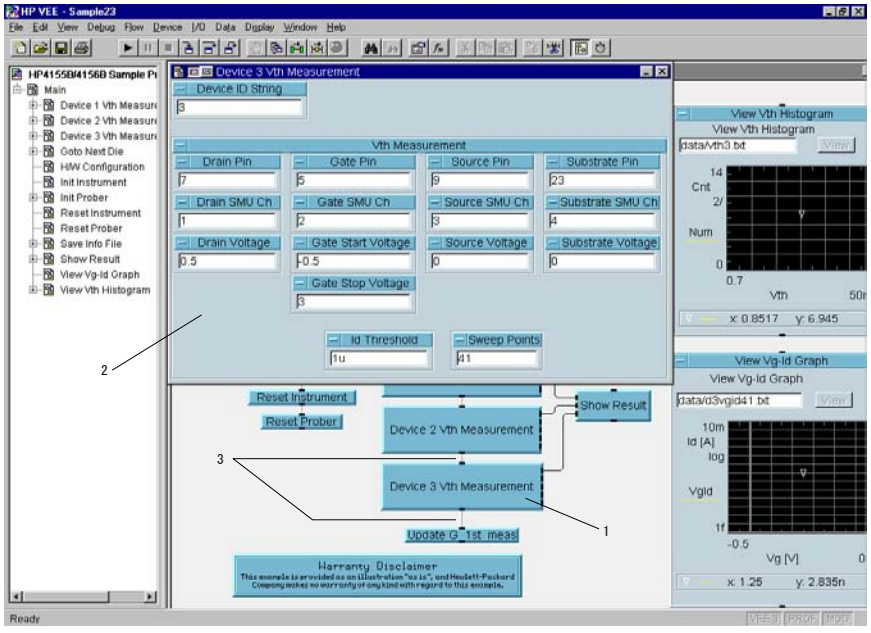


## To Add a Test Device

If your test die includes three MOSFETs, modify the program as shown below. This example modifies sample2.vee, and adds the Vth measurement and display objects for the third MOSFET.

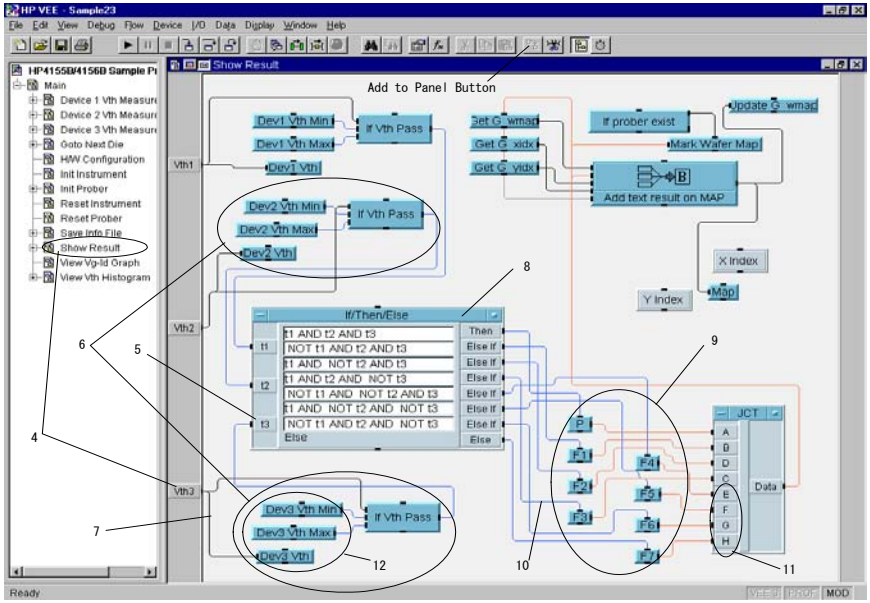
1. Copy and paste the Device 2 Vth Measurement object on the Main program display, and change the title to Device 3 Vth Measurement.
2. Change the measurement setup, Device ID String, pin, voltage, and so on, for the third device on the panel display of the Device 3 Vth Measurement object. See “To Change the Vth Measurement Setup” on page 29.
3. Cut the line between the Device 2 Vth Measurement object and Update G 1st meas object, then connect the lines as shown below.
   - between Device 2 Vth Measurement and Device 3 Vth Measurement
   - between Device 3 Vth Measurement and Update G 1st meas
4. Open the Show Result object program display, and add the Vth3 input terminal. See Figure 7-22.
5. Open the If/Then/Else object, and add the t3 input terminal.

**Figure 7-21** **To Add a Test Device (1)**



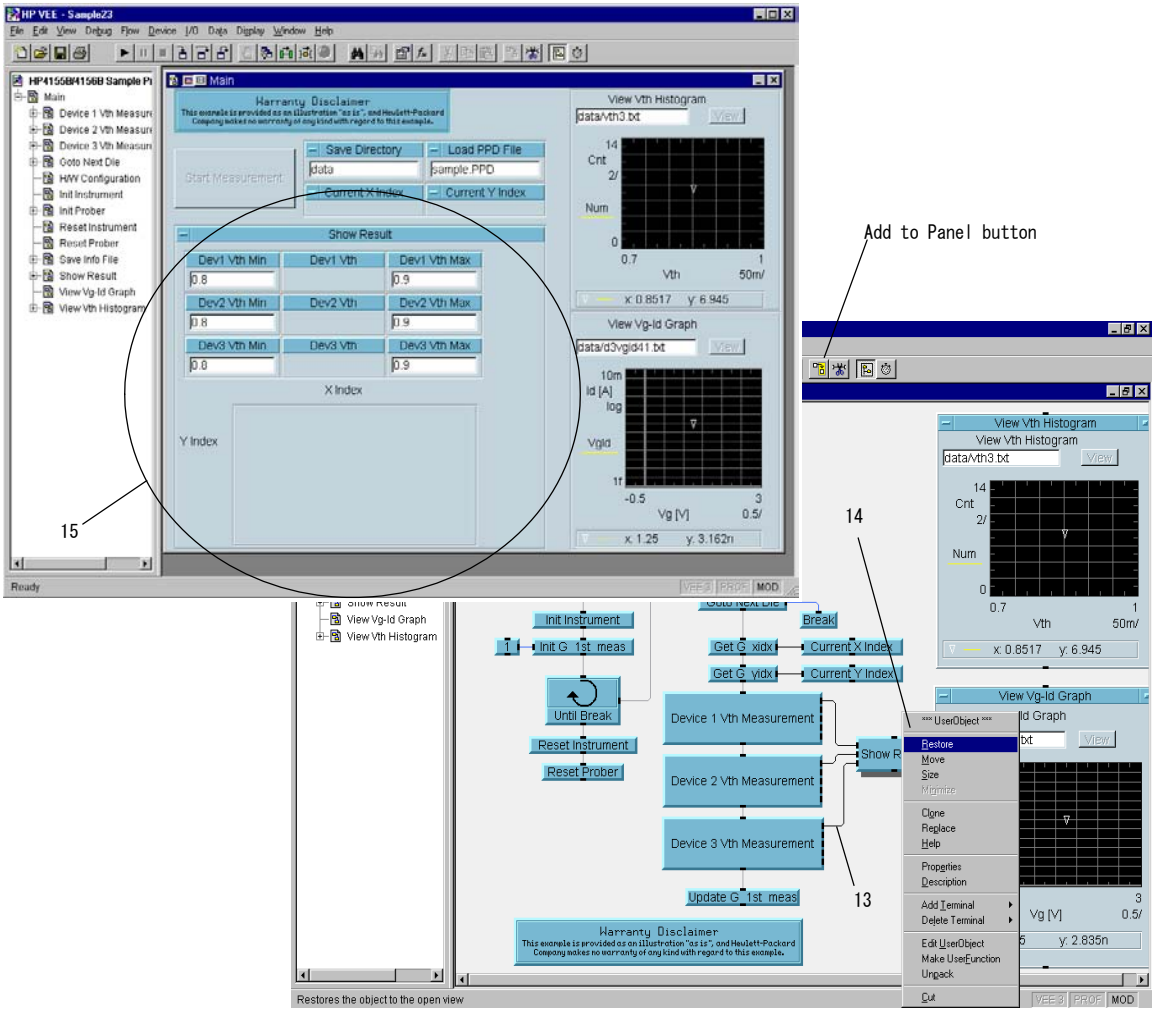
6. Copy and paste the following four objects, and change title Dev2 to Dev3.
   - Dev2 Vth Min, Dev2 Vth Max, Dev2 Vth, and If Vth Pass
7. Connect the lines shown below:
   - between Vth3 terminal and Dev3 Vth object
   - between Vth3 terminal and the data input terminal of If Vth Pass object
   - between If Vth Pass and t3 input terminal of If/Then/Else object
8. Change the definition of the If/Then/Else object as shown in Figure 7-22.
9. Copy and paste P, F1, F2 and F3 objects, and change the title and entry to F4, F5, F6, and F7, respectively.
10. Cut the line between Else and F3, and connect lines for F3, F4, F5, F6, and F7 as shown in Figure 7-22.
11. Open the JCT object, create an additional four input terminals, and connect the lines for E, F, G, and H as shown in Figure 7-22.
12. Open the Dev3 Vth, Dev3 Vth Min, and Dev3 Vth Max objects, and select them. Then click the Add to Panel button. The objects are added to the Show Result panel display. Adjust the position and size of the objects.

**Figure 7-22 To Add a Test Device (2)**



13. Open the Main program display, and connect the line between the Vth terminal of the Dev3 Vth Measurement object and the Vth3 terminal of the Show Result object.
14. Click the right mouse button on the Show Result object, and select the Restore menu. The Show Result object panel display is restored on the Main program display.
15. Select the restored Show Result object, and click the Add to Panel button. The object is added to the Main panel display. Delete the old Show Result object, and adjust the position and size of the new Show Result object.

**Figure 7-23 To Add a Test Device (3)**



**NOTE**

The modification example shown above changes the meaning of the wafer map result display as shown below.

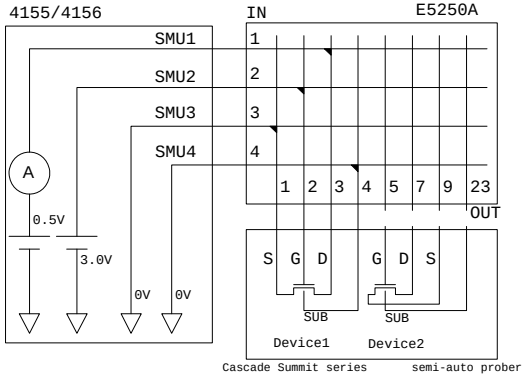
| | |
|---|---|
| P | Test results of all devices are within the allowable range. |
| F1 | Device 1 test result is out of the allowable range. |
| F2 | Device 2 test result is out of the allowable range. |
| F3 | Device 3 test result is out of the allowable range. |
| F4 | Test results of device 1 and 2 are out of the allowable range. |
| F5 | Test results of device 2 and 3 are out of the allowable range. |
| F6 | Test results of device 1 and 3 are out of the allowable range. |
| F7 | Test results of all devices are out of the allowable range. |

## To Add a Measurement Parameter

If you want to add a measurement parameter, such as drain current Id, modify the program as shown below. This example modifies sample2.vee for device 1 only (this example does not modify the objects for device 2).

- Adds the measurement function to the Measurement object.
- Adds the object to set the dummy data to the Meas 4155 (Offline) object.
- Adds the object to set the measurement source to the Vth Measurement object.
- Adds the object to save the measured data to the Device 1 Measurement object.
- Modifies the Show Result object and the Main panel display.

**Figure 7-24** **Id Measurement Setup**



## To Add the Measurement Function and Dummy Data

1. Open the Measurement object using the Agilent VEE program explorer.
2. Create the Vg_spot input terminal in the Measurement object as shown in Figure 7-25.
3. Open the Meas 4155 object, and add the following three functions. Then set the parameters shown using the Edit Function Panel of each function.
   - hp4156b_opc_Q function
   - hp4156b_force function
     a. channel : Use gate_ch variable.
     b. mode : VOLTAGE OUTPUT
     c. range : 0
     d. value : Use Vg variable, and create Vg input terminal.
     e. comp : 1m
     f. polarity : AUTO
   - hp4156b_spotMeas function
     a. channel : Use drain_ch variable.
     b. mode : CURRENT MEASUREMENT
     c. range : 0
     d. value : Use spot_id variable, and create spot_id output terminal.
     e. status : Use spot_id_s variable, and create spot_id_s output terminal.
4. Connect the line between the Vg_spot input terminal and the Vg input terminal of the Meas 4155 object.
5. Add two Flow-Junction (JCT) objects.
6. Create the spot_id and spot_id_s output terminals in the Measurement object.
7. Connect the lines from the spot_id terminal of the Meas 4155 object to the spot_id terminal of the Measurement object via the JCT object.

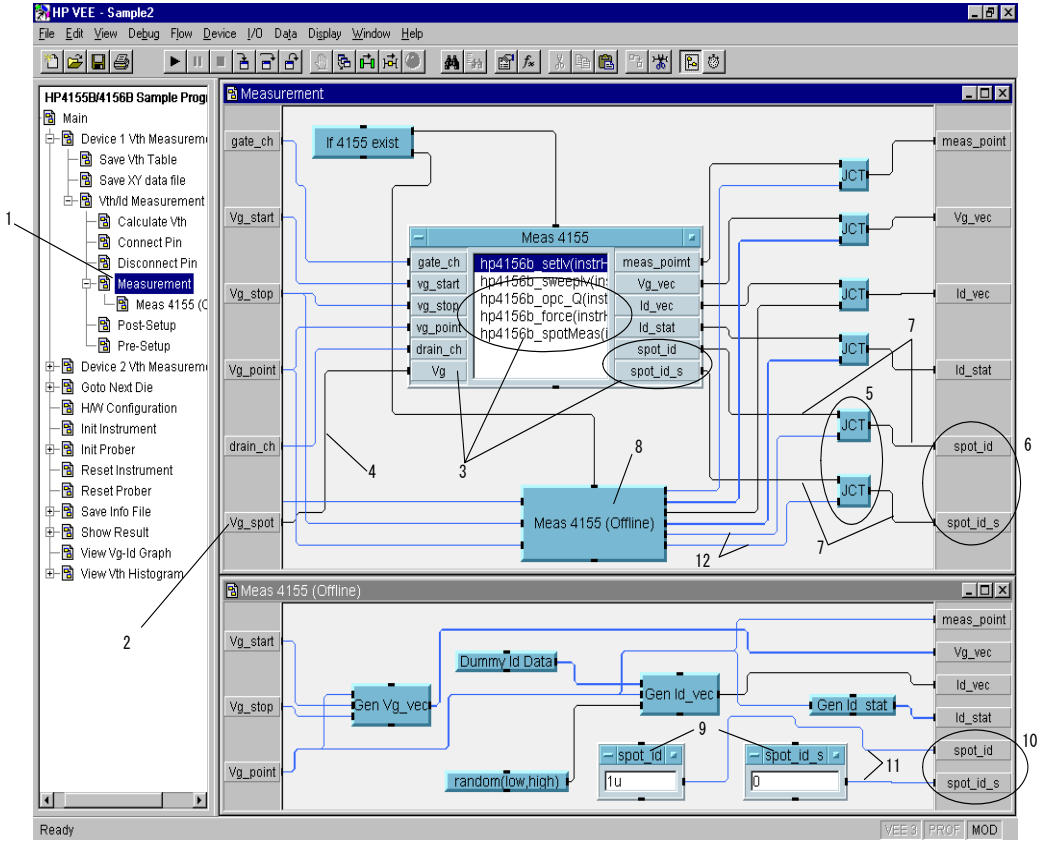
   Then connect the lines from the spot_id_s terminal of the Meas 4155 object to the spot_id_s terminal of the Measurement object via the JCT object.
8. Open the Meas 4155 (Offline) object.

9. Add the Data-Constant-Real and Data-Constant-Integer objects, and set the title to spot_id and spot_id_s, respectively. Then enter “1u” to the entry field of the spot_id object.
10. Create the spot_id and spot_id_s output terminals in the Meas 4155 (Offline) object.
11. Connect the line between the spot_id object and the spot_id terminal.

    Then connect the line between the spot_id_s object and the spot_id_s terminal.
12. Connect the line between the spot_id terminal of the Meas 4155 (Offline) object and the JCT object connected to the spot_id terminal.

    Then connect the line between the spot_id_s terminal of the Meas 4155 (Offline) object and the JCT object connected to the spot_id_s terminal.

**Figure 7-25 To Add the Measurement Function and Dummy Data**

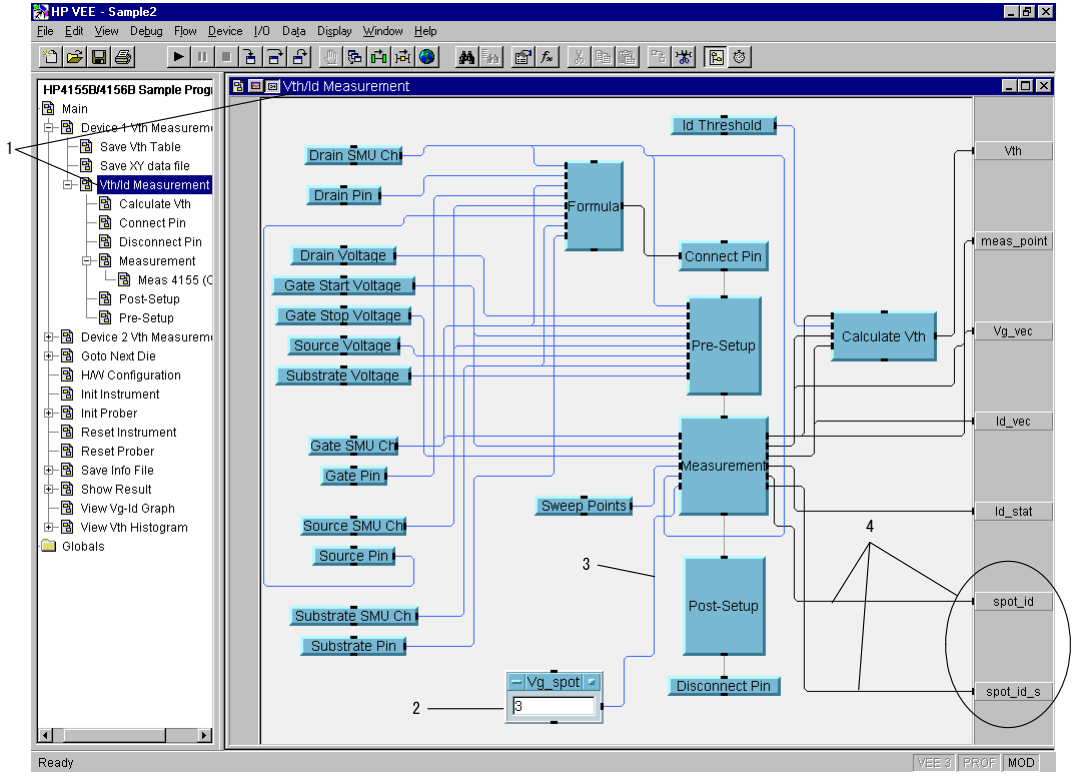


## To Set the Id Measurement Source

1. Open the Vth Measurement object program display, and change the title to Vth/Id Measurement.
2. Add the Data-Constant-Real object, set the title to Vg_spot, and enter any value for gate voltage in volts. This example enters 3.
3. Connect the line between the Vg_spot object and the Vg_spot terminal of the Measurement object.
4. Create the spot_id and spot_id_s output terminals in Vth/Id Measurement object.

   Then connect lines between the spot_id terminal of the Measurement object and the spot_id output terminal, and between the spot_id_s terminal of the Measurement object and the spot_id_s output terminal.

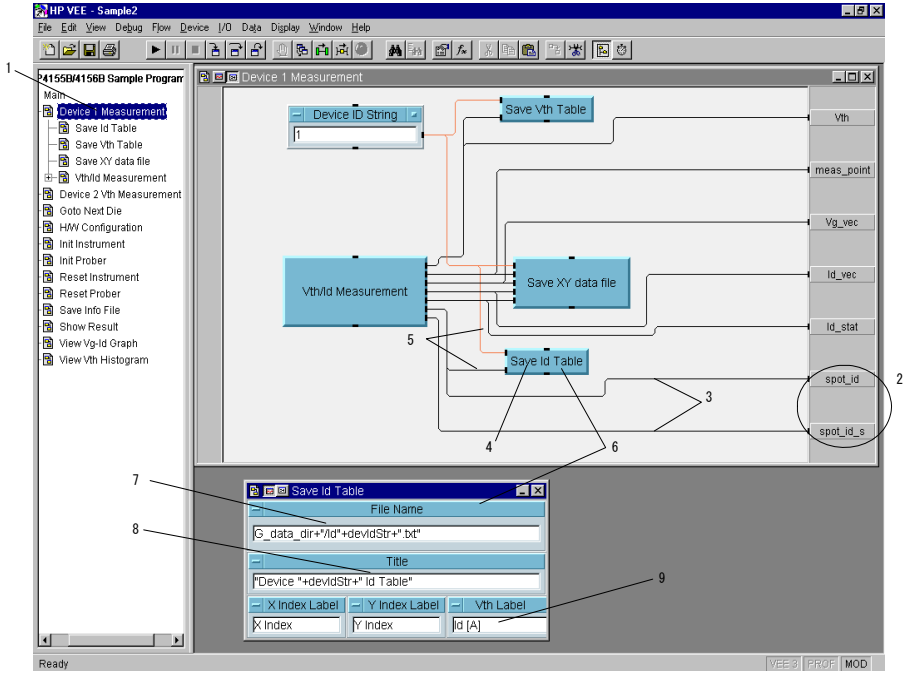
**Figure 7-26** **To Set the Id Measurement Source**



## To Save Measurement Data

1. Open the Device 1 Vth Measurement object program display, and change the title to Device 1 Measurement.
2. Create the spot_id and spot_id_s output terminals.
3. Connect the lines between the spot_id terminal of the Vth/Id Measurement object and the spot_id output terminal, and between the spot_id_s terminal of the Vth/Id Measurement object and the spot_id_s output terminal.
4. Copy and paste Save Vth Table, and change the title to Save Id Table.
5. Connect the lines between the Device Id String object and the devIdStr terminal of the Save Id Table object, and between the spot_id terminal of the Vth/Id Measurement object and the Vth terminal of the Save Id Table object.
6. Open the Save Id Table object panel display.
7. Change the characters *Vth* in the File Name entry field to *Id*.
8. Change the characters *Vth* in the Title entry field to *Id*.
9. Enter Id [A] into the Vth Label entry field.

**Figure 7-27** **To Save Measurement Data**



## To Modify the Show Result and Main Panel Displays

1. Open the Show Result object program display.
2. Add the Display-Alphanumeric object, and set the title to Dev1 Id.
3. Create the spot_id input terminal.
4. Connect the line between the spot_id terminal and the Dev1 Id object.
5. Click the Dev Id object, then click the Add to Panel button. The Dev Id object is added to the panel display of the Show Result object. Adjust the position and size of the object.
6. Open the Main program display, and connect the line between the Show Result object spot_id terminal and the Device 1 Measurement object spot_id terminal.
7. Click the right mouse button on the Show Result object, and select Restore menu. The restored Show Result object is displayed.
8. Click the restored Show Result object, and click the Add to Panel button. The object is added to the Main panel display. Delete the old Show Result object from the panel, and adjust the position and size of the new Show Result object.

**Figure 7-28** **To Modify Show Result and Main Panel**