Model 4200-SCS Semiconductor Characterization System

Reference Manual

4200-901-01 Rev. P / February 2013

4200-901-01

KEITHLEY

A GREATER MEASURE OF CONFIDENCE

Model 4200-SCS
Semiconductor Characterization System
Reference Manual

KTE Interactive Version 9.0

Document Number: 4200-901-01 Rev. P / February 2013

KEITHLEY

Safety Precautions

The following safety precautions should be observed before using this product and any associated instrumentation. Although some instruments and accessories would normally be used with non-hazardous voltages, there are situations where hazardous conditions may be present.

This product is intended for use by qualified personnel who recognize shock hazards and are familiar with the safety precautions required to avoid possible injury. Read and follow all installation, operation, and maintenance information carefully before using the product. Refer to the user documentation for complete product specifications.

If the product is used in a manner not specified, the protection provided by the product warranty may be impaired.

The types of product users are:

Responsible body is the individual or group responsible for the use and maintenance of equipment, for ensuring that the equipment is operated within its specifications and operating limits, and for ensuring that operators are adequately trained.

Operators use the product for its intended function. They must be trained in electrical safety procedures and proper use of the instrument. They must be protected from electric shock and contact with hazardous live circuits.

Maintenance personnel perform routine procedures on the product to keep it operating properly, for example, setting the line voltage or replacing consumable materials. Maintenance procedures are described in the user documentation. The procedures explicitly state if the operator may perform them. Otherwise, they should be performed only by service personnel.

Service personnel are trained to work on live circuits, perform safe installations, and repair products. Only properly trained service personnel may perform installation and service procedures.

Keithley Instruments products are designed for use with electrical signals that are rated Measurement Category I and Measurement Category II, as described in the International Electrotechnical Commission (IEC) Standard IEC 60664. Most measurement, control, and data I/O signals are Measurement Category I and must not be directly connected to mains voltage or to voltage sources with high transient over-voltages. Measurement Category II connections require protection for high transient over-voltages often associated with local AC mains connections. Assume all measurement, control, and data I/O connections are for connection to Category I sources unless otherwise marked or described in the user documentation.

Exercise extreme caution when a shock hazard is present. Lethal voltage may be present on cable connector jacks or test fixtures. The American National Standards Institute (ANSI) states that a shock hazard exists when voltage levels greater than 30V RMS, 42.4V peak, or 60VDC are present. A good safety practice is to expect that hazardous voltage is present in any unknown circuit before measuring.

Operators of this product must be protected from electric shock at all times. The responsible body must ensure that operators are prevented access and/or insulated from every connection point. In some cases, connections must be exposed to potential human contact. Product operators in these circumstances must be trained to protect themselves from the risk of electric shock. If the circuit is capable of operating at or above 1000V, no conductive part of the circuit may be exposed.

Do not connect switching cards directly to unlimited power circuits. They are intended to be used with impedance-limited sources. NEVER connect switching cards directly to AC mains. When connecting sources to switching cards, install protective devices to limit fault current and voltage to the card.

Before operating an instrument, ensure that the line cord is connected to a properly-grounded power receptacle. Inspect the connecting cables, test leads, and jumpers for possible wear, cracks, or breaks before each use.

When installing equipment where access to the main power cord is restricted, such as rack mounting, a separate main input power disconnect device must be provided in close proximity to the equipment and within easy reach of the operator.

For maximum safety, do not touch the product, test cables, or any other instruments while power is applied to the circuit under test. ALWAYS remove power from the entire test system and discharge any capacitors before: connecting or disconnecting cables or jumpers, installing or removing switching cards, or making internal changes, such as installing or removing jumpers.

Do not touch any object that could provide a current path to the common side of the circuit under test or power line (earth) ground. Always make measurements with dry hands while standing on a dry, insulated surface capable of withstanding the voltage being measured.

The instrument and accessories must be used in accordance with its specifications and operating instructions, or the safety of the equipment may be impaired.

Do not exceed the maximum signal levels of the instruments and accessories, as defined in the specifications and operating information, and as shown on the instrument or test fixture panels, or switching card.

When fuses are used in a product, replace with the same type and rating for continued protection against fire hazard.

Chassis connections must only be used as shield connections for measuring circuits, NOT as safety earth ground connections.

If you are using a test fixture, keep the lid closed while power is applied to the device under test. Safe operation requires the use of a lid interlock.

If a  screw is present, connect it to safety earth ground using the wire recommended in the user documentation.

The  symbol on an instrument means caution, risk of danger. The user should refer to the operating instructions located in the user documentation in all cases where the symbol is marked on the instrument.

The  symbol on an instrument means caution, risk of danger. Use standard safety precautions to avoid personal contact with these voltages.

The  symbol on an instrument shows that the surface may be hot. Avoid personal contact to prevent burns.

The  symbol indicates a connection terminal to the equipment frame.

If this  symbol is on a product, it indicates that mercury is present in the display lamp. Please note that the lamp must be properly disposed of according to federal, state, and local laws.

The **WARNING** heading in the user documentation explains dangers that might result in personal injury or death. Always read the associated information very carefully before performing the indicated procedure.

The **CAUTION** heading in the user documentation explains hazards that could damage the instrument. Such damage may invalidate the warranty.

Instrumentation and accessories shall not be connected to humans.

Before performing any maintenance, disconnect the line cord and all test cables.

To maintain protection from electric shock and fire, replacement components in mains circuits - including the power transformer, test leads, and input jacks - must be purchased from Keithley Instruments. Standard fuses with applicable national safety approvals may be used if the rating and type are the same. Other components that are not safety-related may be purchased from other suppliers as long as they are equivalent to the original component (note that selected parts should be purchased only through Keithley Instruments to maintain accuracy and functionality of the product). If you are unsure about the applicability of a replacement component, call a Keithley Instruments office for information.

To clean an instrument, use a damp cloth or mild, water-based cleaner. Clean the exterior of the instrument only. Do not apply cleaner directly to the instrument or allow liquids to enter or spill on the instrument. Products that consist of a circuit board with no case or chassis (e.g., a data acquisition board for installation into a computer) should never require cleaning if handled according to instructions. If the board becomes contaminated and operation is affected, the board should be returned to the factory for proper cleaning/servicing.

Table of Contents

Section 1
Introduction

In this section:

Introduction

This section introduces you to the Keithley Instruments Model 4200-SCS Semiconductor Characterization System (SCS) and its documentation, as follows:

- **Embedded PC policy**: Explains Keithley Instruments' policy concerning the use of third-party software and its no-tamper policy for the Windows® Operating System (OS).
- **Model 4200-SCS system overview**:
  - An overall block diagram and written summary of the system.
  - Basic configurations and capabilities of the Source-Measure Units (SMUs), preamps, pulse generator card, Digital Storage Oscilloscope, and Capacitance-Voltage Units (CVUs) that source and measure the electrical signals that are connected to your Devices Under Test (DUTs).
  - Views and descriptions of the front and rear instrument panels.
  - **Options and accessories**: Provides a description of the options and accessories available for the Model 4200-SCS.
- **Model 4200-SCS documentation overview**: Describes how to use the documentation that is provided with your Model 4200-SCS.

Embedded PC policy

CAUTION **Keithley Instruments warrants the performance of the Model 4200-SCS only with the factory-approved Microsoft® Windows® operating system and application software preinstalled on the Model 4200-SCS by Keithley Instruments. Systems that have been modified by the addition of unapproved third-party application software (software that is not explicitly approved and supported by Keithley Instruments) are not covered under the product warranty. Model 4200-SCS systems with unapproved software may need to be restored to factory-approved condition before any warranty service can be performed (for example, calibration, upgrade, technical support). Services provided by Keithley Instruments to restore systems to factory-approved condition will be treated as out-of-warranty service with associated time and material charges.**

DO NOT reinstall or upgrade the Microsoft Windows operating system (OS) on any Model 4200-SCS. This action should only be performed at an authorized Keithley Instruments service facility. Violation of this precaution will void the Model 4200-SCS warranty and may render the Model 4200-SCS unusable. Any attempt to reinstall or upgrade the operating system will require a return-to-factory repair and will be treated as an out-of-warranty service, including time and material charges.

Although users must not attempt to reinstall or upgrade the operating system, the user can restore the hard drive image (complete with the OS) using the Acronis True Image OEM, described in Section 10 System-level backup and restore software.

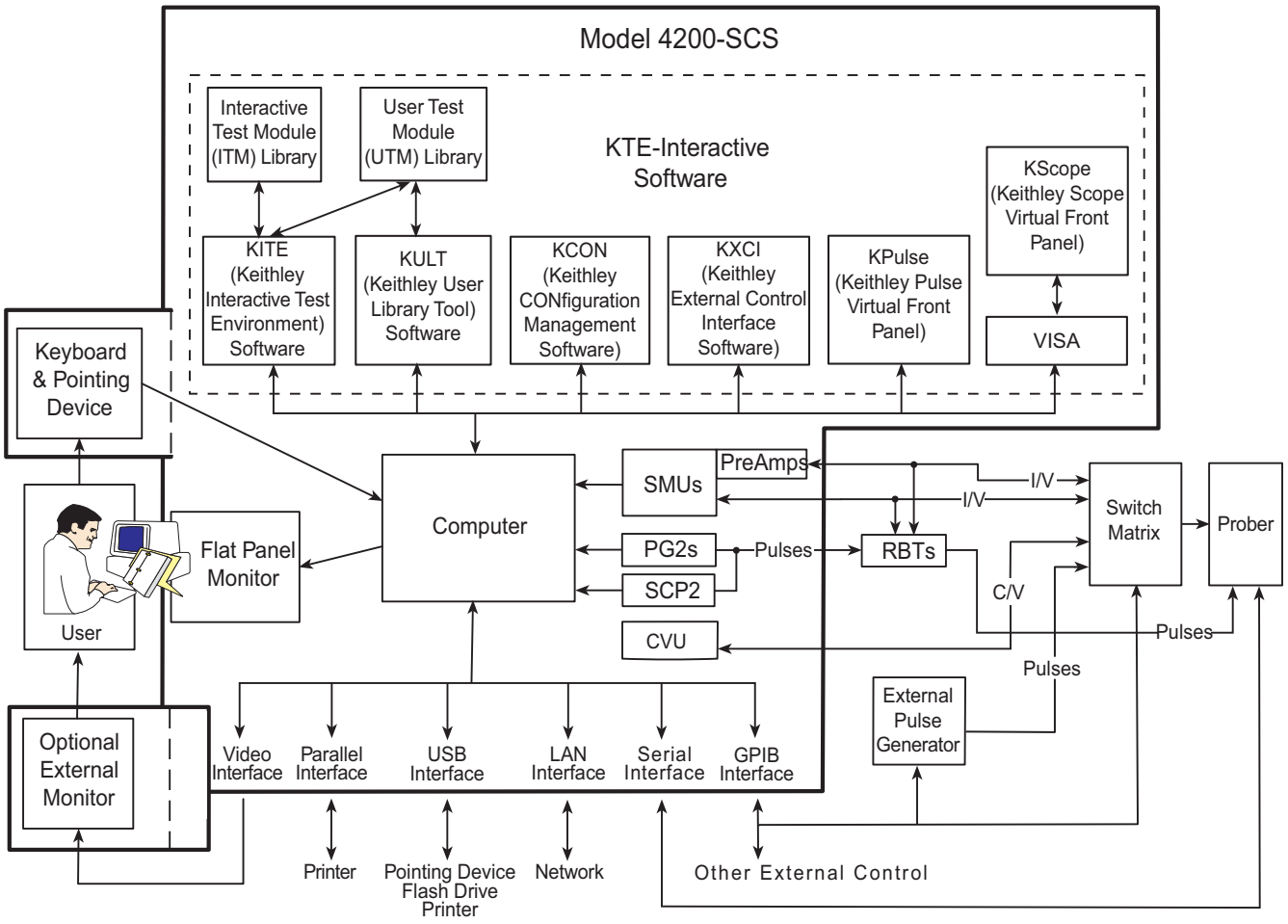
Model 4200-SCS system overview

The Model 4200 Semiconductor Characterization System (4200-SCS) is an automated system that provides I-V, pulsed I-V, and C-V characterization of semiconductor devices and test structures. Its advanced digital sweep parameter analyzer combines speed and accuracy for deep sub-micron characterization.

Tests are easily and quickly configured and executed from the Keithley Interactive Test Environment (KITE). KITE is an application program designed and developed specifically for characterizing semiconductor devices and materials. Source and measurement functions for a test are provided by up to eight source-measure units (SMUs). Test capabilities are extended by support of a variety of external components.

Pulse source and measure tests can be provided by the Model 4225-PMU Ultra-fast IV pulse-measure card. Tests requiring pulse source, but no corresponding pulse measurement, can use the Model 4220-PGU pulse-only card.One typical configuration with pulse source-measure capability would be a Model 4200-SCS system that consists of four SMUs, two Model 4225-PMUs and four Model 4225-RPMs. This system would then have four SMUs and four Pulse IV channels (pulse source and measure), with the RPMs allowing for switching between pulse and SMU test resources.The primary Model 4200-SCS components and typical supported external components are illustrated in Figure 1-1.

Figure 1-1
Model 4200-SCS summary



Software features

The Model 4200-SCS KTE Interactive Software is comprised of six software tools used to operate and maintain the Model 4200-SCS. Each of these tools is described below:

- **KITE***:* Keithley Interactive Test Environment (KITE) is the main Model 4200-SCS device characterization application. KITE is a versatile tool that facilitates both interactive characterization of an individual device or automated testing of an entire semiconductor wafer. Tests are organized into individual projects, which are managed and executed by KITE.
- **KULT***:* The Keithley User Library Tool (KULT) allows test engineers to integrate custom algorithms (user modules) into KITE. Internal Model 4200-SCS instrumentation and external instrumentation can be controlled by means of user modules written using the C programming language. KULT is used to create and manage libraries of user modules.
- **KCON***:* The Keithley CONfiguration (KCON) utility allows test engineers to define the configuration of external GPIB instruments, switch matrices, and analytical probers connected to the Model 4200-SCS. KCON also provides basic diagnostic and troubleshooting functions.
- **KXCI***:* The Keithley External Control Interface (KXCI) allows the use of an external computer to control the SMUs, pulse generator cards, and a scope card remotely over the GPIB (IEEE-488) or Ethernet interface. You can do this in either of two modes: the 4145 emulation mode or the more full-featured 4200 extended mode, which provides access to most of the Model 4200-SCS commands and ranges. KXCI also provides a command set that allows user modules created in KULT to be executed.
- **KPulse***:* KPulse is the Keithley Instruments virtual front panel application that allows direct access to the features of the Keithley pulse cards.
- **KScope***:* KScope is the Keithley Instruments virtual front panel application that allows direct access to the features of the Models 4200-SCP2 or 4200-SCP2HR scope cards.

Hardware features and capabilities

Important hardware features of the Model 4200-SCS mainframe include the following:

- **Eight module slots:** For pulse source-measure, four slots may be used for Keithley pulse cards and one other slot for a scope card.
- **Display:** Built-in 12.1 in. XGA Flat Panel Display (Model 4200-SCS/F) or external monitor with a VGA plug (Model 4200-SCS/C) provides access to the various KTE Interactive Software components. The Model 4200-SCS/F can drive the built-in FPD and an external CRT/monitor simultaneously in either matched resolution or extended desktop mode.
- **Computer:** IBM PC-compatible computer running Microsoft® Windows®.
- **IEEE-488 interface:** Allows the Model 4200-SCS to control GPIB-equipped devices or to be controlled by an external GPIB controller.
- **RS-232 and parallel ports:** Interfaces the unit to peripherals such as a printer, plotter, or prober.
- **Interlock connector**: Interfaces to a test fixture or prober interlock circuit to ensure the instrumentation is controlled in a safe manner.
- **LAN connection:** Built-in Ethernet interface for connections to a local area network.
- **USB Ports:** Four USB 2.0 ports that provide interfaces to peripherals such as pointing devices, printers, scanners, flash drives, external hard drives, and CD-ROM drives.

Key hardware electrical and mechanical characteristics are described in more detail in the next few subsections.

NOTE *Older 4200-SCS systems may differ slightly from the capabilities listed above.*

Source-Measure Hardware

Source-Measure Unit (SMU)

The fundamental instrument module utilized by the Model 4200-SCS is the source-measure unit (SMU). The basic function of a SMU is to perform one of the following source-measure operations:

- Source voltage, measure current voltage
- Source current, measure voltage current

The source of the SMU can be configured to sweep or step voltages or currents, or to output a constant bias voltage or current.

There are two models of source-measure units available. The Model 4200-SMU is a medium power (2W) source-measure unit, and the Model 4210 is a high power (20W) SMU. Table 1-1 lists the maximum limits of the two SMUs.

Table 1-1
Source-measure units

Model	Maximum Voltage	Maximum Current	Maximum Power
4200-SMU	210 V	105 mA	2.2W
4210-SMU	210 V	1.05 A	22W

The Model 4200-SCS can support up to eight Model 4200-SMUs or 4210-SMUs in any combination. The mix of SMUs and preamps installed in the Model 4200-SCS can be customized to address the specific needs of each application.

In general:

- The Models 4210-SMU and 4200-PA modules are optional and can be added.
- Only six SMU modules can be installed if a scope card and one pulse generator card are also installed.

NOTE *May require a hardware upgrade to support more than four 4210-SMU high-power SMUs.*

PreAmp

A Model 4200-PA preamp adds five low current source-measure ranges to an SMU. Without a preamp, the 100nA range (100fA resolution) is the lowest current source-measure range for an SMU. With a preamp installed, the 10 nA, 1 nA, 100 pA, 10 pA and 1pA source-measure ranges are added.

If preamps are ordered, the Model 4200-SCS will be shipped from the factory with the preamps installed on the rear panel of the mainframe. The preamps can be removed from the rear panel and mounted remotely in order to reduce the effects of long cables on measurement quality (long cables can add noise to low current measurements and can cause oscillation with some devices).

An installed preamp is matched to the SMU to which it is connected. Therefore, if you disconnect the preamps to mount them at a remote location, you must ensure that you reconnect each one to its matching SMU.

For details, refer to Source measure unit (SMU) with Model 4200-PA overview in Section 3.

Ground unit (GNDU)

The ground unit on the rear panel of the Model 4200-SCS provides a convenient method of making ground connections. This eliminates the need to use an SMU for this purpose. For details, refer to Ground unit (GNDU) overview in Section 3.

Basic SMU measurement characteristics

Table 1-2 summarizes basic measurement characteristics of the Model 4200-SCS for both the Models 4200-SMU and 4210-SMU, with and without the Model 4200-PA.

Table 1-2
Basic SMU measurement characteristics

Function	4200-SMU	4210-SMU	4200-SMU with 4200-PA	4210-SMU with 4200-PA
Current source ranges (full scale/set resolution)	— — — — — 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA —	— — — — — 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA 1.05 A / 50μA	1.05 pA / 50aA 10.5 pA / 500aA 100.5 pA / 5fA 1.05 nA / 50fA 10.5 nA / 500fA 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA —	1.05 pA / 50aA 10.5 pA / 500aA 100.5 pA / 5fA 1.05 nA / 50fA 10.5 nA / 500fA 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA 1.05 A / 50 μA
Current measurement ranges (full scale/nominal resolution)	— — — — 105 nA / 1pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA /10nA 10.5 mA / 100nA 105 mA / 1 μA — —	— — — — — 105 nA / 1pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1nA 1.05 mA / 10nA 10.5 mA / 100nA 105 mA / 1 μA 1.05 A / 10 μA	1.05 pA / 10aA 10.5 pA / 100aA 100.5 pA / 1fA 1.05 nA / 10fA 10.5 nA / 100fA 105 nA / 1pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1nA 1.05 mA / 10nA 10.5 mA / 100nA 105 mA / 1 μA —	1.05 pA / 10 aA 10.5 pA / 100 aA 100.5 pA / 1 fA 1.05 nA / 10 fA 10.5 nA / 100 fA 105 nA / 1 pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA / 10 nA 10.5 mA / 100nA 105 mA / 1 μA 1.05 A / 10 μA
Voltage source ranges (full scale/set resolution)	210 mV/5 μV 2.1V/50 μV 21V/500 μV 210 V/5 mV	210 mV/5 μV 2.1V/50 μV 21V/500 μV 210 V/5 mV	210 mV/5 μV 2.1V/50 μV 21V/500μV 210 V/5 mV	210 mV/5 μV 2.1V/50 μV 21V/500 μV 210 V/5 mV
Voltage measurement ranges (full scale/nominal resolution)	210 mV/1 μV 2.1V/10 μV 21V/100 μV 210 V/1 mV	210 mV/1 μV 2.1V/10 μV 21V/100 μV 210 V/1mV	210 mV/1μV 2.1V/10 μV 21V/100 μV 210 V/1mV	210 mV/1μV 2.1V/10 μV 21V/100 μV 210 V/1mV
Power	2.2W	22W	2.2W	22W

Pulse source-measure hardware

Keithley Instruments' pulse source-measure hardware for the Model 4200-SCS includes one or more pulse Keithley pulse cards and a scope card (Model 4200-SCP2 or 4200-SCP2HR).

Capacitance (CVU) hardware

Keithley Instruments' capacitance-voltage hardware for the 4200-SCS includes up to one capacitance-voltage unit (CVU). Refer to Multi-Frequency C-V measurements (Model 4200-CVU) in Section 15 for details.

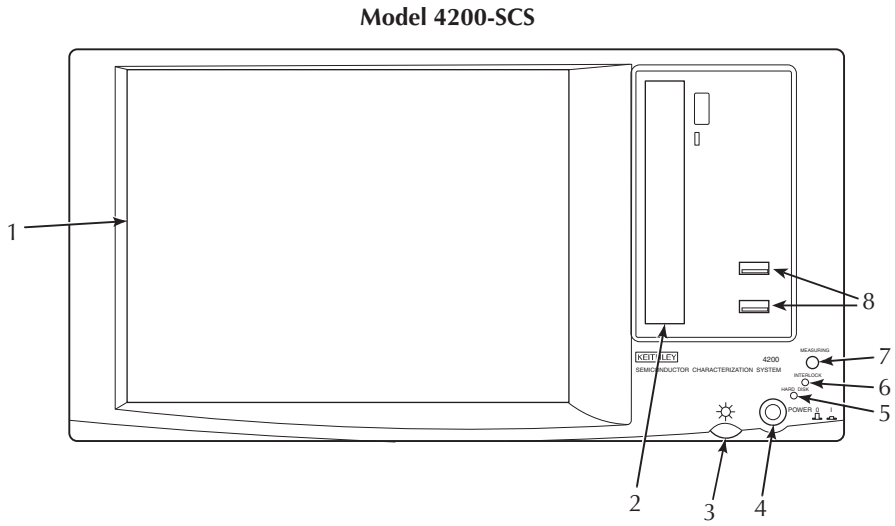
Instrument panels

All operator interfaces are on the front panel of the Model 4200-SCS, and all connection interfaces are on the rear panel. The next two subsections describe the front and rear panels.

Front panel

Figure 1-2 shows the front panel of the Model 4200-SCS. The various components are summarized below the figure.

Figure 1-2
Front panel

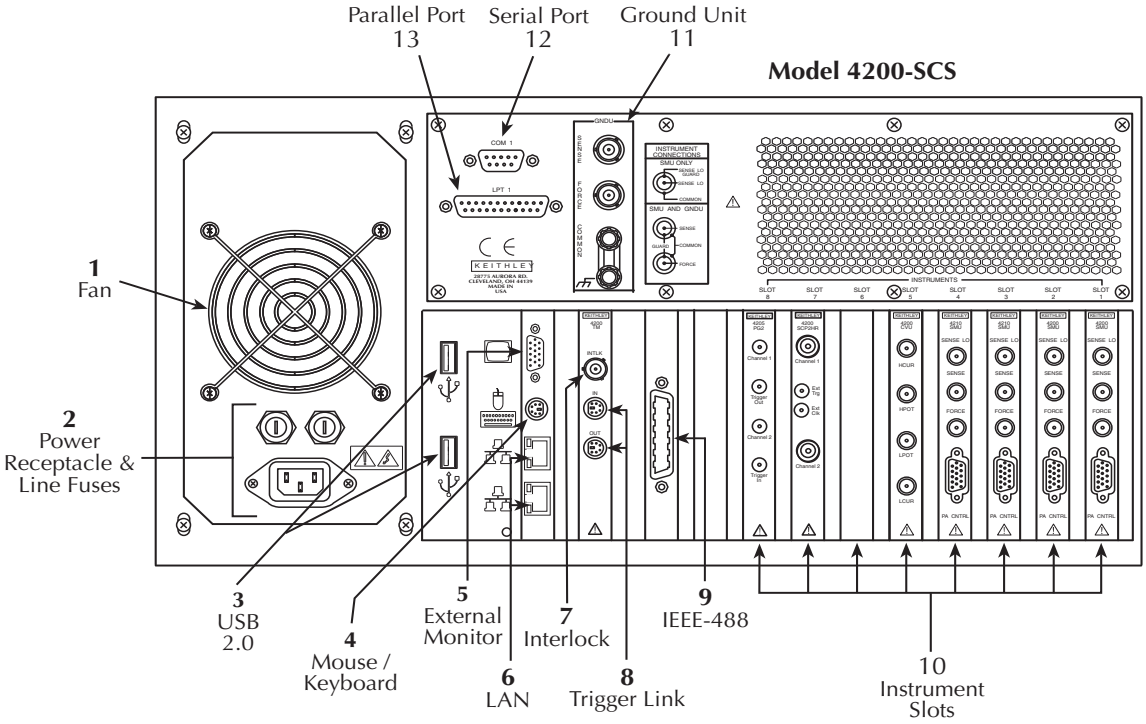


1. Display — Displays graphical user interface, data, graphs, and system operation information. Note: Model 4200-SCS/C has no display and requires an external CRT/monitor.
2. DVD/CD-R/W drive — Provides a means to install or update system software, manuals, and utilities.
3. Display brightness — Allows you to set the FPD display to the desired brightness, and turn off the FPD backlight.
4. POWER switch — Turns main system power on or off.
5. HARD DISK indicator — Turns on when the hard disk is being accessed.
6. INTERLOCK indicator — Turns on when the test fixture interlock is closed.
7. ACTIVE indicator — Turns on when any internal cards are energized.
8. Two v2.0 USB ports — Interfaces to peripherals (for example, pointing devices, printers, scanners, flash drives, external hard drives, and CD-ROMs).

Rear panel

Figure 1-3 shows the rear panel of the Model 4200-SCS mainframe. The various components are summarized below the figure.

Figure 1-3
Rear panel



1.	Fan	Provides system cooling.
2.	Power receptacle and line fuses	Connects to line power through supplied line cord. Two line fuses protect the unit.
3.	2 v2.0 USB connectors	Interfaces to peripherals (for example, pointing devices, printers, scanners, flash drives, external hard drives, and CD-ROM drives).
4.	Mouse and keyboard connector	Included Y-Cable to connect the mouse or other pointing device and the system keyboard (see Figure 2-1).
5.	External monitor port	Used to connect an external CRT or other monitor.
6.	2 LAN connectors	Interfaces the unit to an ethernet local area network.
7.	Interlock connector	Connects to test fixture or prober safety interlock.
8.	Trigger link connectors	**For factory use only.**
9.	IEEE-488 connector	Connects to peripherals or computer with GPIB interface.
10.	Instrument slots	Any of the eight slots can be used for a SMU. Pulse generator cards are installed starting in slot 8 and continuing to the right. A scope card can be installed in the slot next to the last pulse generator card. A CVU card will be located just after the last SMU. In Figure 1-3, a pulse generator card is installed in slot 8 and a scope card is installed in slot 7. SMUs are installed in slots 1 through 4.
11.	Ground unit	Provides a convenient way to make system-level COMMON and SENSE circuit connections.
12.	Serial port	Connects to RS-232 peripherals, such as a prober.
13.	Parallel port	Used to interface to printer or other parallel device.

NOTE *The actual rear panel layout may vary slightly from the diagram shown in Figure 1-3.*

NOTE *LAN2 connection may be deactivated by default. Go to Windows device manager to enable. That is, START; Control Panel; Administrative Tools; Computer Management; Network Adapters; <Right Mouse> Enable.*

Options and accessories

SMU options

Model 4200-SMU: Medium-power source-measure unit; 2W, 200 V, 100 mA (factory installed; not field-upgrade able)

Model 4210-SMU: High-power source-measure unit; 20W, 200 V, 1A (factory installed; not field-upgrade able)

Model 4200-PA: Remote preamp; extends the low-current capability of a Model 4200-SMU or 4210-SMU by adding the following five ranges: 1 pA, 10 pA, 100 pA, 1 nA, and 10 nA (factory installed; not field-upgrade able)

Compiler option

Microsoft Visual Studio: The C++ compiler is used to create, modify, and debug KULT modules (user library and user modules). When ordered with a Keithley Model 4200-SCS system, the software is installed at the factory.

Pulsing: Source and measure options

Model 4220-PGU: Dual channel pulse generator. Includes the following:

- Model 4220-PGU pulse generator card
- Four SMA to SMA cables (male to male, 2 meters)
- Two Triax to SSMC cables
- Two Y-cables

Model 4225-PMU: Dual channel pulse/measure card. Includes the following:

- Model 4225-PMU pulse/measure card
- Four SMA to SMA cables (male to male, 2 meters)
- Two Triax to SSMC cables
- Two Y-cables

NOTE *A Model 4200-SCS chassis may have a maximum total of six Model 4220-PGUs and Model 4225-PMUs. For example, a chassis that has six Model 4225-PMUs and one Model 4220-PGU is not supported. The number of active channels in a test is limited by the Model 4200-SCS power supply (for additional information, see "Model 4200-SCS power supply limitations" on page 16-73).*

Model 4225-RPM: Remote Pulse (and switch) Module. Includes the following:

- One Model 4225-RPM Remote Pulse (and switch) Module

- One Model 4200-MAG-BASE (for mounting the 4225-RPM)
- One White RPM communication/measurement cable (2 meters)
- Two SMA to SMA cables (male to male, 8 inches)
- Two SMA female to BNC male connectors
- Two BNC female to 3-lug triax male connectors
- Four Triax protective caps
- Two SMA protective caps

Model 4200-BTI-A: Ultra-Fast BTI Bundle. Includes the following:

- One Model 4225-PMU pulse/measure card
- Two Model 4225-RPM remote pulse (and switch) modules
- One ACS Software Package (Automated Characterization Software)

Model 4200-PMU-PROBER-KIT: Includes the following:

- Two BNC male to BNC male 1.5 meter cables
- Two BNC female to SMA male connectors
- Two SMA Tees (female-male-female)
- Two SMA to SMA cables (male to male, 8 inches)
- Two Micro-Triax to SMA (no guard) connectors
- Three BNC female to 3-lug Triax male connectors
- Three BNC female to BNC female connectors
- Four SMA female to BNC male connectors
- One BNC shorting cap

Model 4200-PIV-A: Pulse I-V solution bundle. Includes the following:

- Model 4205-PG2 (includes everything listed for the pulse generator)
- Model 4200-SCP2 (includes everything listed for the scope)
- Two Model 4205-RBT Bias Tee adapters
- Model 8101-PIV test fixture
- Model 4200-PIV software
- Cables and connectors

Model 4200-PIV-HR: Pulse I-V solution bundle. Includes the following:

- Model 4205-PG2 (includes everything listed for the pulse generator)
- Model 4200-SCP2HR (includes everything listed for the scope)
- Two Model 4205-RBT Bias Tee adapters
- Model 8101-PIV test fixture
- Model 4200-PIV software
- Cables and connectors

Model 4200-PIV-Q: Quiescent Point Pulse I-V solution bundle. Includes the following:

- Three Model 4205-PG2s (includes everything listed for each pulse generator)
- Model 4200-SCP2HR (includes everything listed for the scope)
- Model 8101-PIV test fixture
- Model 4200-PIV-Q software
- Cables and connectors

Model 4200-FLASH: Flash memory testing solution bundle. Includes the following:

- Two Model 4205-PG2s (includes everything listed for each pulse generator)
- Model 4200-FLASH software

- Cables and connectors

Model 4200-SCP2-ACC: ZTEC ZT500PRB-00 BNC probe

Capacitance: Source/Measure Options

Model 4200-CVU: Capacitance-Voltage Unit includes the following:

- One Model 4200-CVU capacitance/voltage card (CVU)
- Four SMA-to-SMA cables (100-ohm, male-to-male, 1.5 meters, Red)
- Four SMA-female-to-BNC-male connectors
- Two BNC Tee connectors (female-male-female)
- One 4200-CVU Quick Start Guide

Model 4200-CVU-PROBER-KIT: Capacitance-Voltage Unit Prober Kit includes the following:

- Four SMA-to-SMA cables (100-ohm, male-to-male, 3 meters, Red)
- Four Triax-to-BNC connectors (triax 3-lug male to BNC 2-lug female)
- Four Triax-to-BNC connectors (triax 3-lug male to BNC 2-lug female, with Guard Removed)
- Four Triax-to-Triax barrels (female to female)
- Four BNC-to-BNC (female to female)
- Four SMA-female-to-BNC-male connectors
- Two SMA-Tees (female-male-female)
- Two SMA-to-SSMC Dual Prober connectors with local ground

Service and calibration options

Model 4200-3Y-REPAIR: Model 4200-SCS 3-year return repair service

Model 4200-5Y-REPAIR: Model 4200-SCS 5-year return repair service

Model 4200-3Y-CAL: Model 4200-SCS 3-year calibration service

Model 4200-5Y-CAL: Model 4200-SCS 5-year calibration service

Model 4200-CAL: Model 4200-SCS calibration service

Model 4200-CERT: ANSI NCSL-Z540 calibration certification with test data

Model 4200-UPGRADE: Model 4200-SCS upgrade service; includes installation of new instruments, preamps, calibration, and verification

Model 4200-OS-UPGRADE: Model 4200-SCS Windows operating system and KTEI software upgrade; includes new hard drive and additional RAM, Windows 7, and the latest version of KTEI system software.

Model 4200-COMPLETE-REFURB: Model 4200-SCS upgrade service; includes new embedded single board computer, power supply, hard drive, RAM, 9-slot backplane, Windows 7, and the latest version of KTEI system software.

Computer accessories

Model 4200-MOUSE: Microsoft ambidextrous 2-button mouse

Remote preamp mounting accessories

Model 4200-MAG-BASE: Magnetic base to mount the Model 4200-PA or 4205-RBT remotely on the platen of a probe station

Model 4200-VAC-BASE: Vacuum base to mount the Model 4200-PA or 4205-RBT remotely on the platen of a probe station

Model 4200-TMB: Triaxial mounting bracket to mount the Model 4200-PA or 4205-RBT remotely on the triaxial feed-through connectors of a probe-station dark box

Other accessories

Model 4200-MAN: Printed manual set for the Model 4200-SCS (the manual set on CD-ROM is included with the Model 4200-SCS base unit)

Model 4200-CART: Mobile cart that provides system portability for the Model 4200-SCS mainframe

Model 8101-TRX: Transistor test fixture

Model 8101-PIV: Pulse I-V test fixture

Model 8007: Semiconductor test fixture

Switch matrices

Model 4200-UL-LS-XX series: An ultra-low current switch matrix, available with 12 to 72 pins; the XX in the part number specifies the number of pins in 12-pin increments

Model 4200-UL-RS-XX series: An ultra-low current and remote sense switch matrix, available with 6 to 30 pins; the XX in the part number specifies the number of pins in 12-pin increments

Model 4200-LC-LS-XX series: A cost-effective low current switch matrix, available with 12 to 72 pins; the XX in the part number specifies the number of pins in 12-pin increments

Model 4200-GP-RS-XX series: A general-purpose switch matrix that supports remote sense on 12 to 72 pins; the XX in the part number specifies the number of pins in 12-pin increments

Cabinets and mounting accessories

Model 4200-CAB-20UX: 20U cabinet (35 in.)

Model 4200-CAB-25UX: 25U cabinet (44 in.)

Model 4200-CAB-34UX: 34U cabinet (60 in.)

Model 4200-RM: Slide rack-mounting kit for the Models 4200-SCS/F and 4200-SCS/C

Model 4200-CRT-RM: Fixed rack-mounting kit for the Model 4200-CRT

Model 4200-KEY-RM: Slide rack-mounting kit for the keyboard and pointing device

Model 2288: Rack-mounting kit for the Model 4200-590 C-V Analyzer

Cables

NOTE *All Model 4200-SCS systems and instrument options are factory-supplied with required cables.*

Model 4200-RPC-X series: Remote preamp control cable connects the Model 4200-PA to a Model 4200-SMU or 4210-SMU. Available in lengths of 0.3m (1.0 ft), 2m (6.5 ft), 3m (9.8 ft), or 6m (19.7 ft). The X in the part number specifies the length.

Model 4200-TRX-X series: Ultra-low noise preamp triax cable is used to connect a Model 4200-PA to a test fixture; terminated at both ends with 3-slot male triax connectors. Available in lengths

of 0.3m (1.0 ft), 1m (3.3 ft), 2m (6.5 ft), and 3m (9.8 ft). The X in the part number specifies the length.

Model 4200-MTRX-X series: Miniature triax cable, 2m (6.5 ft), is used to connect a Model 4200-SMU or Model 4210-SMU to a test fixture; is equipped with a miniature male triax connector on one end and a standard 3-slot male triax connector on the other end. Available in lengths of 1m (3.3 ft), 2m (6.5 ft), and 3m (9.8 ft). The X in the part number specifies the length.

Model 236-ILC-3: Interlock cable, 3m (9.8 ft), is used to connect the mainframe interlock connector to the test fixture or prober interlock.

Model 7007-X series: Shielded GPIB cables that are used to connect the Model 4200-SCS mainframe to the GPIB (IEEE-488) bus; use shielded cables and connectors to reduce electromagnetic interference (EMI). The Model 7007-1 is 1m (3.3 ft) long; the Model 7002-2 is 2m (6.5 ft) long.

Model 7078-TRX-BNC: 3-lug triax-to-BNC adapter.

Model 4200-SCS documentation overview

The Model 4200-SCS documentation, comprised of a Quick Start Manual, Applications Manual, and Reference Manual, is overviewed below. Guidance for using the documentation is also provided.

Surveying the documentation

The organization and content of the Model 4200-SCS documentation is surveyed below.

User's Manual synopsis

The User's Manual sections are organized as shown in Table 1-3.

Table 1-3
User's Manual synopsis

Section number	Section title	Description
1	Getting Started	Covers basic start-up information including; 1) installation and system connections; 2) a hardware overview; 3) connecting DUTs; and 4) how to run a basic test.
2	Model 4200-SCS Software Environment	Explains the various aspects of the software environment, including; 1) understanding KITE and the project structure; 2) ITMs versus UTMs and how to create an ITM; 3) basic test execution; 4) displaying and managing test results, and 5) KITE library management.
3	Common Device Characterization Tests	Explains how to; 1) perform an I-V test; 2) perform a C-V test; 3) perform a pulsed I-V test; 4) perform a quiescent-point pulsed I-V test; 5) perform reliability (stress-measure) tests; 6) perform AC stress (WLR); 7) perform a flash memory test; 8) perform charge pumping; and 9) perform a charge trapping test.
4	How to Control Other Instruments with the Model 4200-SCS	Explains how to; 1) control a switch matrix; 2) control a probe station; 3) control an external pulse generator; and 4) control an external CV analyzer
5	How to Generate Basic Pulses	Explains how to use the Keithley Pulse Application (KPulse) to generate standard pulse waveforms, Segment ARB® waveforms, and custom file arb waveforms (full-arb).
6	How to Control the 4200-SCS Remotely from a PC	Explains control using the Ethernet or the GPIB. Also, introduces you to the KXCI control interface which is used with the GPIB.

Table 1-3 (continued)
User's Manual synopsis

Section number	Section title	Description
7	How to Create UTMs Using C Programming	Explains how to use the Keithley User Library Tool (KULT) to create UTMs. Includes tutorials to 1) create new a new user library and user module, 2) create a user module that returns data arrays, and 3) call one user module from within another.

Reference Manual synopsis

The Reference Manual is organized as shown in Table 1-4.

Table 1-4
Reference Manual synopsis

Section number/ letter	Section title	Description
1	Introduction	Overviews hardware/software features and characteristics, surveys and guides use of the documentation, and lists available options and accessories for the Model 4200-SCS.
2	Installation	Covers unpacking and inspection, system connections, basic SMU connections, and power and environmental requirements. Provides a brief summary of the pulse generator card and scope card.
3	Source-measure hardware	Describes the Model 4200-SCS source-measure hardware in more detail, including the following: • Physical: Front and rear panel interfaces. Instrument terminals and connectors for the Model 4200-SCS mainframe, Models 4200-SMU, 4210-SMU and 4200-PA (preamp). • Needed services and environmental conditions. Mounting options for the preamp. • Electrical: Block diagrams, operating characteristics, and basic connections for the Models 4200-SCS mainframe, Models 4200-SMU, 4210-SMU, and 4200-PA (preamp).
4	Connections and configuration	Provides details for basic source-measure connections, external test equipment connections, and control and data connections.
5	Source-measure concepts	Describes various source-measure concepts, including guarding, sensing, sink operation, source-measure configurations, and sweeping. Covers how to make stable measurements, and low-current measurements, and discusses possible sources of interference.
6	Keithley Interactive Test Environment (KITE)	Explains and illustrates the characteristics and operation of KITE. Some topics covered include the Project Navigator, ITMs, and UTMs. Explains how to run tests and view and analyze collected data.
7	Keithley CONfiguration Utility (KCON)	Explains how to use this utility to add components (switch matrix, test fixture, C-V analyzer, pulse generators, and probe station) to the test system. Also provides diagnostic and troubleshooting information for the Model 4200-SCS.
8	Keithley User Library Tool (KULT)	Explains how to use this tool to create and manage libraries of user modules.
9	Keithley External Control Interface (KXCI)	Explains how to use an external controller (PC) to control (by means of GPIB or Ethernet) the SMUs, pulse generator cards, and a single scope card of the Model 4200-SCS remotely. It also explains how to execute user modules created in KULT remotely.

Table 1-4 (continued)
Reference Manual synopsis

Section number/ letter	Section title	Description
10	System administration	Explains basic system administration operations. These include software upgrades, interfacing with networks and printers, limiting system access, and other operations.
11	Pulse source-measure concepts	Provides documentation for using Keithley pulse cards and a Model 4200-SCP2 or 4200-SCP2HR (digital storage oscilloscope card) to perform pulse source-measure tests.
12	Pulse projects	Provides the documentation for the Keithley Instruments pulse projects. Describes the tests (ITMs and UTMs) for each project.
13	KPulse	The KPulse graphical user interface (GUI) is used to control the Keithley pulse cards. This section has been moved to Section 5 of the User's Manual.
14	KScope	Explains how to use the graphical user interface (GUI) that is used to control the Model 4200-SCP2 or 4200-SCP2HR digital storage oscilloscope.
15	Multi-Frequency C-V Measurements	Provides an overview of the 4200-CVU Capacitance-Voltage unit and how AC impedance measurements are made. All measurement functions and parameters are explained as well as DC Bias and sweep characteristics. Cables, connections, and compensation tools and techniques are also explained in detail. Considerations for making good C-V measurements through a switch matrix are explained. Finally, a number of common C-V tests are explained, complete with test set-up parameters and expected output data.
16	Models 4220-PGU, 4225-PMU, and 4225-RPM	Provides details on using the PGU and PMU pulse cards, and the RPM (remote pulse and switch module.
A	Creating project prompts	Explains how to add pop-up windows to a KITE project to prompt the user to perform an action. You create these prompts by means of UTMs that connect to the user modules of the `Winlib` user library. It also documents the project plans provided for these products.
B	Using switch matrices	Provides a tutorial to add a switch matrix to the test system. Also provides signal connection information and discusses key matrix concepts. Describes the user modules in the `matrixulib` user library used to control the Keithley Instruments Model 707/708 Switch Matrices.
C	Using a Keithley Instruments Model 590 C-V Analyzer	Provides a tutorial to add each C-V Analyzer to the test system. Also provides signal connection information and discusses key concepts. Describes the user modules in the `ki590ulib` and `hp4980 ulib` user libraries that are used to control the Keithley Instruments 590 and Agilent 4980 C-V meters.
D	Using an Agilent Model 4980A LCR Meter	
E	Using a Keithley Instruments Model 82 C-V System	Covers the following: key measurement concepts, instrument connections, system configuration, use of Keithley-supplied projects, measurement-parameter setup, user-library reference information, and simultaneous C-V analysis principles.
F	Using an Agilent Model 8110A/81110A Pulse Generator	Provides a tutorial to add the pulse generator to the test system. Also provides signal connection information and discusses key concepts. Describes the user modules in the `hp8110ulib` that are used to control the Agilent 8110A pulse generator. Also documents the user library for the Agilent 81110 pulse generator.

Table 1-4 (continued)
Reference Manual synopsis

Section number/ letter	Section title	Description
G	Using a probe station	Appendices G through L explain how to add a probe station to the test system. They also provide connection information and discuss key concepts. Describes the user modules in the `prbgen` user library that are used to control the probe station(s).
H	Suss MicroTec Model PA-200 Prober	
I	Micromanipulator 8860 Prober	
J	Using a manual or Fake prober	
K	Cascade Summit-12000 Prober	
L	Signatone CM500 Prober	
M	WLR testing	Includes background information on HCI degradation, summaries for using the Model 4200-SCS project plans, and supplemental information for performing the JEDEC standard procedures for V-Ramp and J-Ramp.
N	Additional user libraries	Documents Model 4200-SCS user libraries not covered in other reference sections.
O	Advanced applications	Explains how to 1) control a switch matrix, 2) sequence tests on multiple devices, and 3) customize a user test module (UTM).

Other documentation

Your Model 4200-SCS documentation also includes:

- **Application notes:** Practical examples of how to use the Model 4200-SCS and other related products to perform application-specific tasks.
- **Data sheets:** The Model 4200-SCS technical data sheet and other related product data sheets.

The application notes and data sheets, as well as the Quick Start Manual, Applications Manual, and Reference Manual, may be accessed in PDF format from the Model 4200-SCS Complete Reference. The Model 4200-SCS Complete Reference is preinstalled on your system and can be accessed using Microsoft Internet Explorer, which also comes preinstalled on your system. The Complete Reference on your system resembles a website, but actually resides in your instrument. Keithley Instruments also provides a copy of the Model 4200-SCS Complete Reference on a CD-ROM that comes with each Model 4200-SCS.

Distinguishing special text items in the manuals

Italic, bold, and upper-case letters, the Courier font, quotation marks, and special characters distinguish certain text items from the general text in this manual. The following text conventions are used (exclusive of heading styles):

- **10 point Arial Bold** distinguishes the following:
  - All user-interaction items within a KTE Interactive or Windows® Graphical User Interface (GUI): Project and Configuration Navigator Components, commands, screen messages, menu names, menu selections, and dialog-box items, including captions, user selections, and typed user inputs (however, the GUI title is not boldfaced, but it is capitalized as on the screen).
  - ***CAUTION*** statements
- *10 point Arial Italic* distinguishes the following:

  - Emphasis in general
  - *NOTE* statements
- 10 POINT ARIAL UPPER CASE distinguishes keyboard keys, such as ENTER and CTRL.
- `10 point Courier` distinguishes the following:
  - Software code statements and C-functions
  - **Calc** worksheet functions
  - **Formulator** functions
  - Command-line commands
  - User-module and user-library names
  - Directory paths
- Double quote marks distinguish the following:
  - Cross references to sections/chapters in the manuals, such as Keithley User Library Tool (KULT) in Section 8.
  - Cross references to sections/chapters in other documents (the name of the other document, not in quotes, accompanies the section/chapter name).
  - Project and Configuration Navigator Components such as default.
  - Literals, such when referring to the 5 V labels on I/O connectors.
- Double-quote marks and **10 Point Arial Bold** distinguishes text that is both Project and Configuration Navigator Component and user-interaction items within a KTE Interactive or Windows® Graphical User Interface (GUI).
- Dual angle brackets, < >, typically enclose a generic, descriptive name for an essential parameter, function, file, directory path, or other item. in a command definition. In practice, the generic, descriptive name must be replaced with a real name.
- Dual rectangle brackets, [ ], typically enclose a generic, descriptive name for an *optional* parameter, function, file, directory path, or other item. in a command definition. In practice, the generic, descriptive name must be replaced with a real name.

Section 2
Installation

In this section:

Introduction

This section contains information about handling and installing the Keithley Instruments Model 4200-SCS Semiconductor Characterization System:

- **Unpacking and inspection:** Covers unpacking and inspection for damage. Lists the items shipped with every unit. Provides instructions for returning the unit to Keithley Instruments should it become in need of repair.
- **System connections:** Explains how to connect the keyboard (and optional mouse), probe station, printer, and LAN to the Model 4200-SCS.
- **SMU connections:** Describes the simplest method to make SMU connections to the device under test (DUT).
- **Pulsing: Source and measure hardware**: Provides a brief summary of Keithley Instruments modules (cards) used for pulsing and waveform capture.
- **Power requirements:** Covers line power requirements for the Model 4200-SCS, and shows how to connect the power line cord.

CAUTION **When you start one of the KTE Interactive software tools for the first time, you will be required to respond affirmatively to an on-screen license agreement before proceeding further. If you do not respond with a "Yes" answer, your system will be nonfunctional until you reinstall the software.**

NOTE *The condensed installation information in this section is intended to get your Model 4200-SCS set up and ready to turn on as quickly as possible. Detailed information on connections is provided in "Connection considerations" in Section 4.*

Unpacking and inspection

Inspection for damage

After unpacking the mainframe, carefully inspect the unit for any shipping damage. Report any such damage to the shipping agent, as such damage is not covered by the warranty.

Shipment contents

The following items are included with the Model 4200-SCS:

- Model 4200-SCS Semiconductor Characterization System with any ordered source-measure units (SMUs) factory-installed
- Ordered Model 4200-PA modules factory-installed
- Ordered Keithley pulse cards factory-installed
- Ordered Model 4200-SCP2 or 4200-SCP2HR Digital Storage Oscilloscope card factory-installed
- Ordered Model 4200-SCP2-ACC scope probe (BNC)
- Ordered pulse application packages (refer to Pulsing: Source and measure options in Section 1)

- Cables, connectors, adapters and other accessories that are supplied with the pulse generator, scope, and pulse application packages. Pulsing: Source and measure options in Section 1 lists the supplied accessories for the pulsing options
- Line cord
- Model 4200-SCS Quick Start Manual
- Miniature triaxial cables, two per Model 4200-SMU or 4210-SMU, 2m (6 ft)[1]
- Triaxial cables, two per Model 4200-PA, 2m (6 ft)
- Interlock cable
- Keyboard with integrated pointing device and Y-Cable
- System software and manuals on CD-ROM
- Microsoft® Windows®
- Ordered Microsoft® Visual Studio® factory-installed

Manual package

System manuals are provided on a CD-ROM and are preinstalled on the hard drive. If a complete set of printed manuals is required, order the optional manual package, Keithley Instruments Model Number 4200-MAN. The manual package includes any pertinent addenda. Because the manuals are provided in PDF format, they can be printed from any computer that is connected to a printer by using Adobe Acrobat Reader.

Repacking for shipment

Should it become necessary to return the Model 4200-SCS for repair, carefully pack the entire unit in its original packing carton or the equivalent, and follow these instructions:

- Call Keithley Instruments' repair department at 1-888-KEITHLEY (1-888-534-8453) for a Return Material Authorization (RMA) number.
- Let the repair department know the warranty status of the Model 4200-SCS Semiconductor Characterization System.
- Write ATTENTION REPAIR DEPARTMENT and the RMA number on the shipping label.
- Complete and include the Service Form located at the back of this manual.

1. Not included when SMU is ordered with a Model 4200-PA.

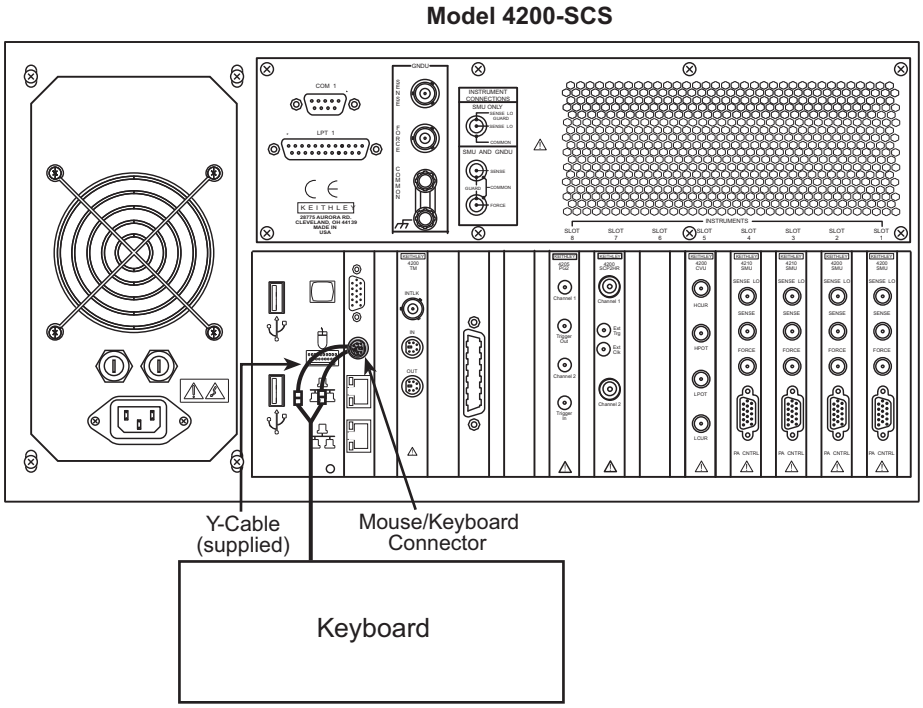
System connections

Connecting the keyboard and mouse (optional)

The cable for the keyboard is terminated with two connectors. One connector is for the keyboard functions and the other is for the integrated pointing device. Figure 2-1 shows the connections to the Model 4200-SCS.

If you wish to use an optional mouse, unplug the connector for the pointing device and connect your mouse.

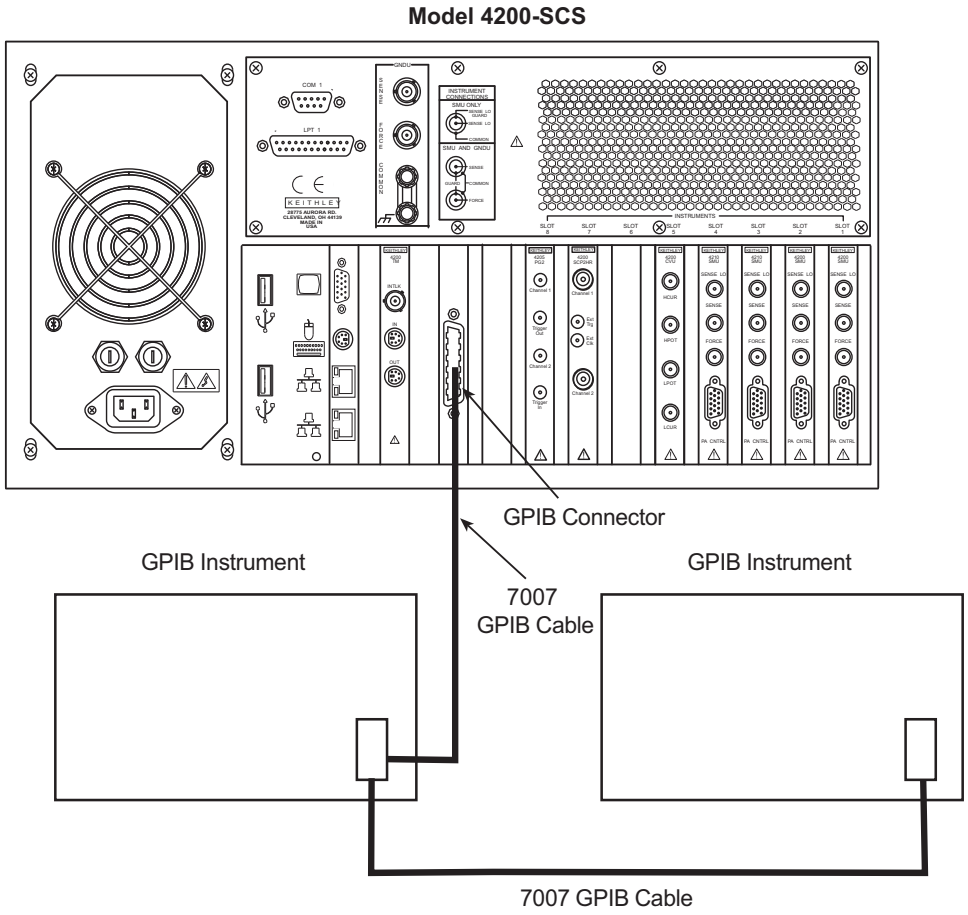
Figure 2-1
Keyboard connections



Connecting GPIB instruments

The Model 4200-SCS can control one or more external instruments by way of the IEEE-488 General Purpose Instrument Bus (GPIB). An example of typical instruments used in a test system with the Model 4200-SCS are a switch matrix and a C-V meter. Figure 2-2 shows how to connect GPIB instruments to the Model 4200-SCS.

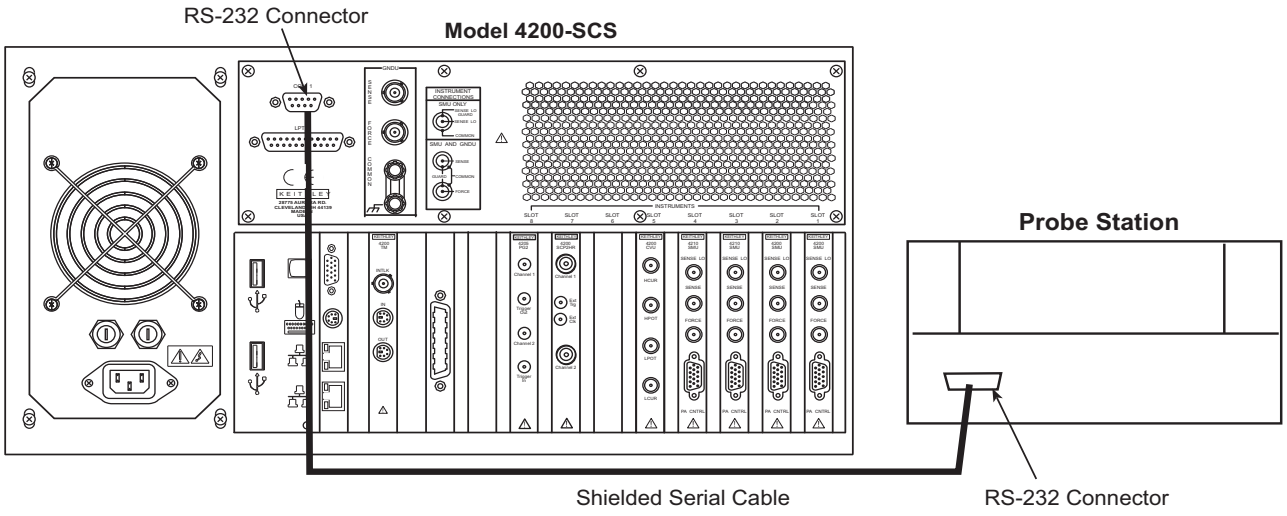
Figure 2-2
GPIB instrument connections



Connecting a probe station

A probe station can be controlled over the RS-232 interface and is connected to the Model 4200-SCS, as shown in Figure 2-3.

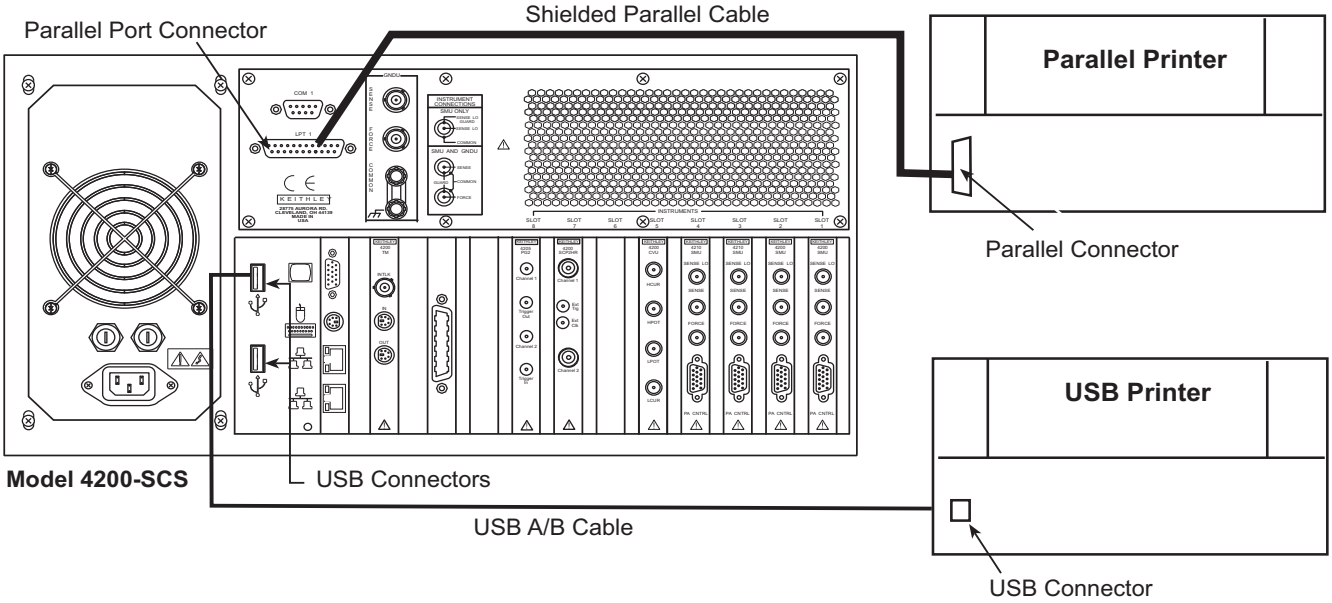
Figure 2-3
Probe station connections



Connecting a printer

As shown in Figure 2-4, a printer can be connected to the parallel port of the Model 4200-SCS. If you are using a USB printer, connect it to one of the v2.0 USB connectors.

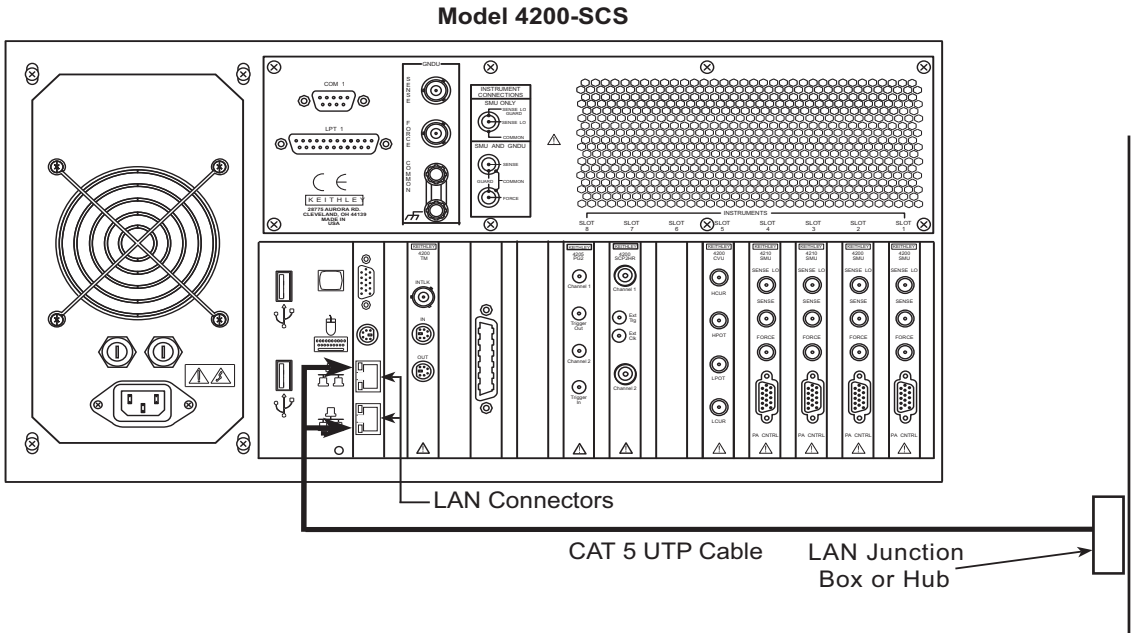
Figure 2-4
Printer connections



Connecting a LAN

The two LAN connectors on the Model 4200-SCS are standard RJ-45 connectors intended for use with UTP (Unshielded Twisted Pair) cable. For best results, use only CAT 5 UTP cables equipped with RJ-45 connectors to connect your LANs, as shown in Figure 2-5.

Figure 2-5
LAN connections



SMU connections

The following information explains how to connect the source-measure units (SMUs) to the device under test (DUT):

WARNING ***Do not touch test cables or connectors when powering up the Model 4200-SCS. Hazardous voltage may be output momentarily, posing a safety hazard that could result in personal injury or death.***

Do not turn on the Model 4200-SCS until you have reviewed the safe power-up procedure described later in this section under the heading, Powering the Model 4200-SCS.

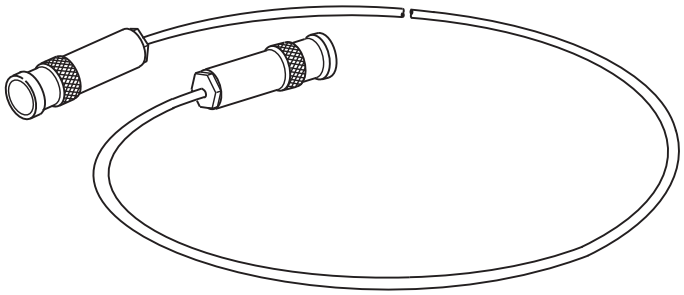
CAUTION ***Do not connect the DUT to the Model 4200-SCS before powering it up, because the hazardous voltage that may be output momentarily at power-up could damage the DUT.***

If optional preamps were ordered with your Model 4200-SCS, they have already been installed on the rear panel. All tests should be performed using the preamps, because the installed SMUs were optimized at the factory to use them.

Triax cables

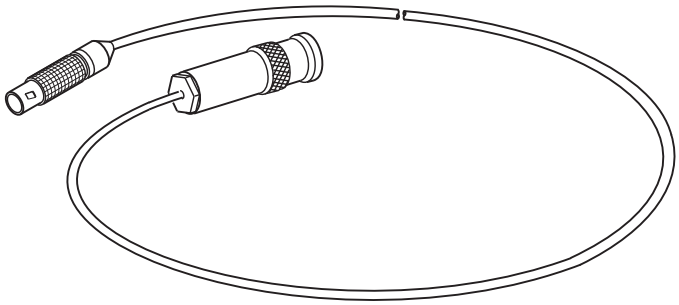
Triax cables are supplied to make connections to the DUT (device under test). With preamps installed, use the low noise triax cables, which are terminated with 3-slot triax connectors on both ends. One end of the cable connects to the preamp and the other end connects to the DUT test fixture or probe station.

Figure 2-6
Triax cable Model 4200-TRX-X



If your system does not have preamps installed, use the cables that have a miniature triax connector on one end and a standard 3-slot triax connector on the other end. The cable end terminated with the miniature connector connects directly to the SMU, and the other end connects to the test fixture or probe station.

Figure 2-7
Triax cable Model 4200-M TRX-X



CAUTION **With PreAmps installed, NEVER make connections directly to any of the miniature triax connectors on the SMU modules; this may result in damage to the SMU or DUT or may produce corrupt data.**

Basic connections

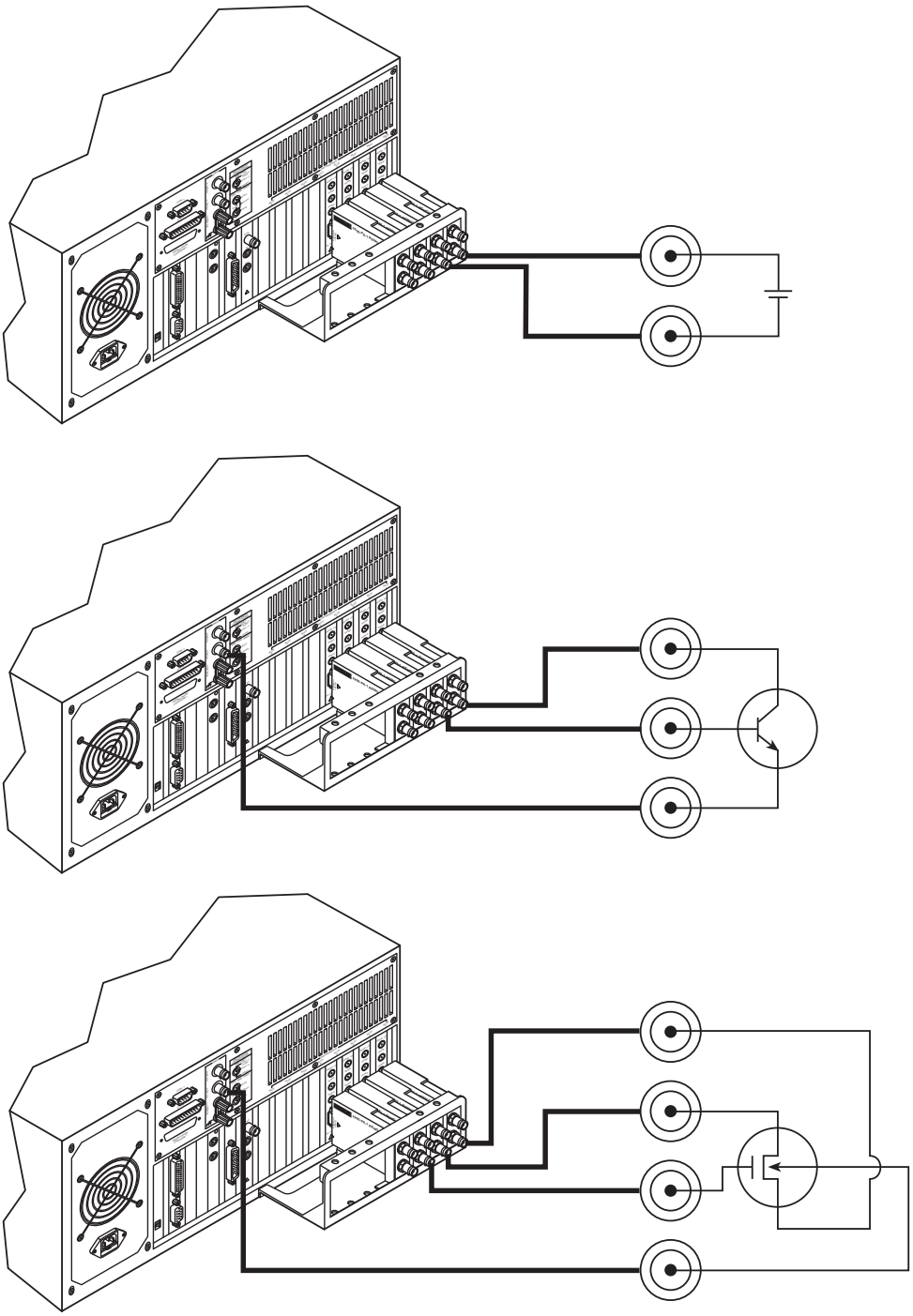
The simplest method to connect SMUs to the DUT is to use one SMU for each terminal of the device. When setting up a test, the FORCE terminal (center conductor) of the SMU is used to apply voltage or current to the device. The FORCE terminal or ground unit can also be used to connect the device terminal to the COMMON circuit.

NOTE *Complete details on connections (including SENSE terminal connections) are provided in Section 4, Connections and Configuration.*

Figure 2-8 shows SMU connections to 2-terminal, 3-terminal, and 4-terminal devices. Notice that only the FORCE HI terminal of the SMUs is connected to the device terminals. FORCE HI is the center conductor of the triax cable.

CAUTION **Connecting the SMU or ground unit SENSE terminal without the FORCE terminal may damage the instrument and give erroneous results.**

Figure 2-8
SMU connections to DUT



Ground unit (GNDU)

A device terminal can be connected directly to the SMU circuit COMMON at the ground unit (GNDU) on the rear panel of the Model 4200-SCS (see Figure 2-9). The ground unit has four connectors:

SENSE: The center conductor of this triax connector is connected directly to the common SENSE LO signal that is shared by all installed SMUs.

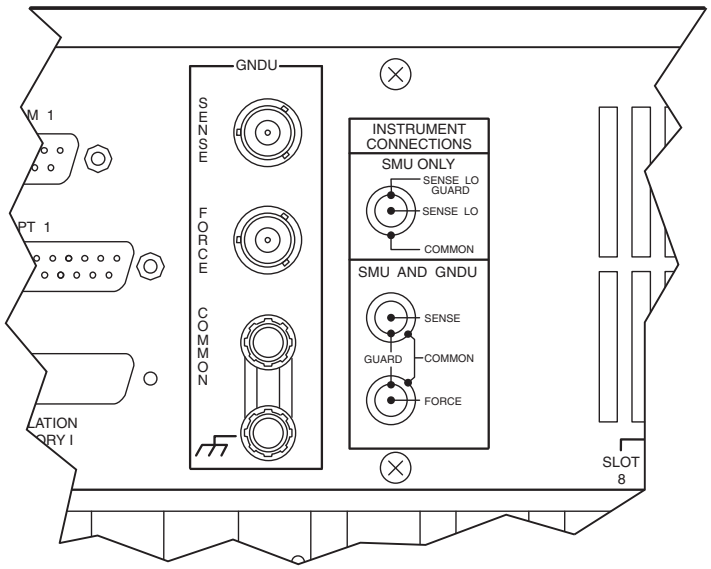
FORCE: The center conductor of this triax connector is connected directly to the SMU circuit COMMON and is rated for 2.6 A maximum.

COMMON: This banana jack, or binding post, is also connected directly to the circuit COMMON and is rated for 9 A maximum.

Chassis ground: This banana jack, or binding post, is connected directly to chassis ground. Note that with the ground link installed, the COMMON circuit is connected to the chassis ground.

For details, see Using the ground unit in Section 4.

Figure 2-9
Ground unit (GNDU) connectors

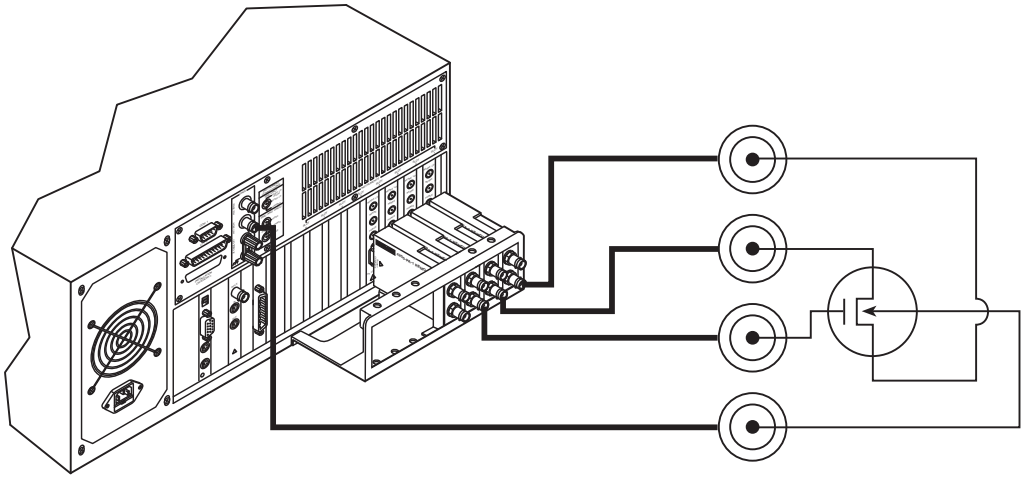


The most obvious use of the ground unit is to make circuit COMMON connections in a system that does not have enough SMUs for all device terminals. Another reason to use the ground unit is the higher current capability of the ground unit connectors.

When a SMU is used to connect a device terminal to circuit COMMON, current is limited to either 105 mA (Model 4200-SMU) or 1.05 A (Model 4210-SMU). If a ground current for your test exceeds 1.05 A, you should connect the device terminal directly to the ground unit. The FORCE triax connector can safely handle up to 2.6 A while the COMMON banana jack can handle up to 9 A.

Figure 2-10 shows how a device terminal can be connected to the ground unit.

Figure 2-10
Signal common connection using ground unit (GNDU)



Test fixtures

There are two types of test fixtures for the Model 4200-SCS: low voltage fixtures (less than ±20 volts) and high voltage (greater than ±20 volts). High voltage fixtures require extra precaution to ensure there are no shock hazards. Whenever the interlock of the Model 4200-SCS is asserted, the FORCE and GUARD terminals of the SMUs and preamps should be considered high voltage, even if they are programmed to a non-hazardous voltage current.

Testing with less than ±20 volts

For testing discrete devices, a test fixture equipped with 3-lug triax connectors is necessary to allow the Model 4200-SCS to be connected to the discrete device. Figure 2-11 shows a basic test fixture to test a two-terminal device. The test fixture's exterior enclosure should be constructed of metal, and the metal should be connected to COMMON. The DUT should be mounted on test terminals that are insulated using a material that has high resistivity such as Teflon. Guarding will improve the quality of the measurement by reducing leakage and parasitic capacitance. The Keithley Instruments "Low Level Measurements Handbook" provides an in-depth discussion on guarding and other techniques that are useful for building quality test fixtures. Contact a Keithley Instruments sales or service office to obtain a copy.

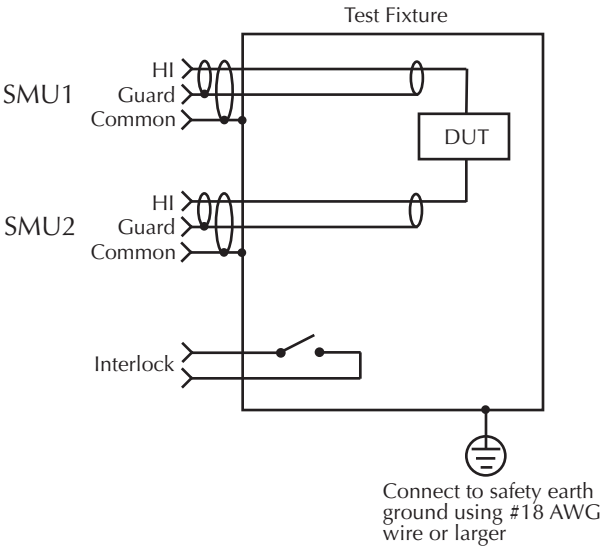
NOTE *The Model 4200-SCS will function on all current ranges and up to ±20 volts without the interlock being asserted. The maximum voltage on the SMU and PreAmp terminals is not hazardous when the interlock is not asserted.*

Testing with more than ±20 volts

If voltages greater than ±20 volts are needed for testing, then the above step should be followed and the following additional steps are required. An interlock switch must be added to the fixture to ensure that hazardous voltages are not present when the fixture's exterior enclosure is open and to enable the Model 4200-SCS to output higher voltages when the fixture's exterior enclosure is closed. In addition, the exterior enclosure must be connected to COMMON and/or safety ground using #18AWG wire or greater. Care must be taken to ensure that the wiring (FORCE, GUARD, and SENSE) within the fixture does not electrically contact the exterior enclosure. For more details on the Model 4200-SCS interlock system, see Control and data connections in Section 4.

WARNING ***Asserting the interlock will allow the SMU and PreAmp terminals to become hazardous, possibly exposing the user to high voltage that could result in personal injury or death. SMU and PreAmp terminals should be considered hazardous even if the outputs are programmed to be low voltage. Precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 volts, Category I.***

Figure 2-11
Typical test fixture



Mounting PreAmps in a probe station

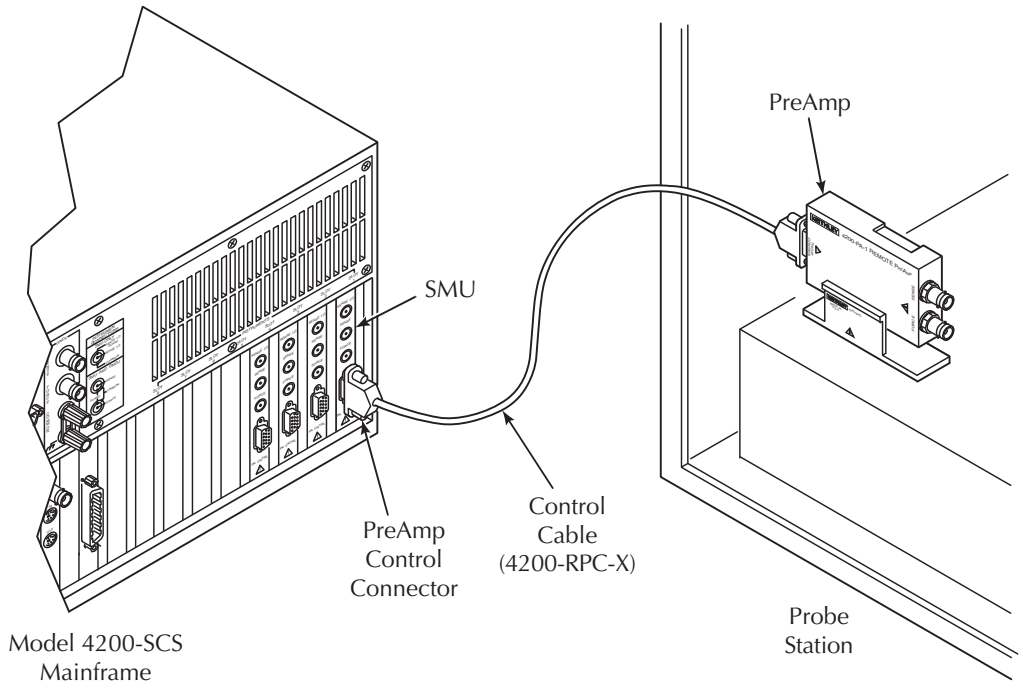
The preamps can be mounted remotely on a probe station using one of the optional mounting kits (see Options and accessories in Section 1). Follow the general steps below to mount and connect a preamp on a probe station remotely (refer to Figure 2-12). Details are provided in the packing list provided with the mounting kit.

NOTE *A PreAmp is matched to the SMU it is connected to. Therefore, when you disconnect the PreAmps to mount them in a probe station, you must make sure to reconnect each one to its matching SMU.*

1. Turn off the power for the Model 4200-SCS mainframe.
2. Disconnect the preamps from the rear panel of the Model 4200-SCS. They are secured to the rear panel by a mounting bracket.
3. Mount the preamp at the desired remote location using the appropriate mounting kit.
4. Connect the control/power cable between the preamp Control connector on the preamp, and the PA CNTRL connector on the SMU.
5. Ensure that the connecting cable is secure at both ends.

For details on preamp mounting, see PreAmp mounting in Section 3.

Figure 2-12
Installing a PreAmp on the probe station



Pulsing: Source and measure hardware

Keithley Instruments has additional instrumentation designed for pulsing, including the Model 4205-PG2 pulse generator, and the Model 4200-SCP2 or 4200-SCP2HR digital storage oscilloscope (DSO). Up to four pulse generator cards and one scope card can be used in a Model 4200-SCS test system.

Pulse generator cards and a scope card are preinstalled at the factory when you order them with your initial Model 4200-SCS purchase. The pulse generator card and scope card can also be ordered separately or as part of a pulse package (for instance, the Model 4200-PIV-HR Pulse I-V solution bundle). Pulse packages include the hardware and any special components required for its intended applications. Refer to Pulsing: Source and measure options in Section 1 for details on all available pulsing options.

Pulse generators can be used along with DSO for AC (pulse) applications or integrated into a system with Keithley Instruments SMUs to include DC source and measure capability.

Keithley pulse card: The dual-channel pulse generator provides voltage pulses as short as 20ns in high speed mode or up to ±20 V (into 50 Ω) in high voltage mode. In addition to standard pulse output, it also provides arb generator functions: full-arb or Segment ARB® waveform.

Model 4200-SCP2 or **Model 4200-SCP2HR**: The dual-channel DSO performs measurements in both the time (frequency, rise/fall time) and voltage domains (amplitude, peak-peak, and so on.). The two scopes are similar, except the Model 4200-SCP2HR has a higher resolution. Also included is KScope, which is a graphical user interface (GUI) to configure and control the scope card.

NOTE *Complete details on the pulse generator card and scope card are provided in Section 11.*

Environmental requirements

Shipping and storage environment

To avoid possible damage or deterioration, the Model 4200-SCS should be shipped and stored within the following environmental limits:

- Temperature: -10°C to +60°C
- Relative humidity: 5% to 90%, non-condensing

Operating environment

Temperature and humidity

The Model 4200-SCS should be operated within the following environmental limits:

- Temperature: +15°C to +40°C
- Relative humidity: 5% to 80%, non-condensing

NOTE *SMU and PreAmp accuracy specifications are based on operation at 23°C ±5°C and between 5% and 60% relative humidity. See the product specifications for additional temperature and humidity derating factors outside these ranges.*

Proper ventilation

To avoid over-heating, the Model 4200-SCS should be operated in an area with proper ventilation. Allow at least eight inches of clearance at the back of the mainframe to assure sufficient airflow.

CAUTION **To prevent damaging heat buildup and other harmful environmental conditions and to ensure specified performance, adhere to the following precautions:**

- **Keep the venting holes and fan free of dust, dirt, and contaminants, so that the unit's ability to dissipate heat is not impaired.**
- **Keep the fan vents and cooling vents from becoming blocked.**
- **Do not position any devices that force air (heated or unheated) adjacent to the unit into cooling vents. This additional airflow could compromise accuracy performance.**
- **When rack-mounting the unit, make sure there is adequate airflow around the sides, bottom, and back to ensure proper cooling.**
- **Rack mounting high power dissipation equipment adjacent to the Model 4200-SCS could cause excessive heating to occur.**
- **To ensure proper cooling in rack situations with convection cooling only, place the hottest equipment (that is, power supply) at the top of the rack. Precision equipment, such as the Model 4200-SCS, should be placed as low as possible in the rack where temperatures are the coolest. Adding spacer panels below the unit will help ensure adequate airflow.**

CAUTION **A large system (for example, multiple SMUs, multiple pulse generators, and a scope) draws more power than a small system. Therefore, the internal power supply will generate more heat. However, it is imperative that a system of any size have proper ventilation. Even for a small system, inadequate ventilation could cause heat to build up and cause damage.**

Cleanliness

To avoid internal dirt buildup that could degrade performance and affect longevity, the Model 4200-SCS should be operated in a clean, dust-free environment.

Powering the Model 4200-SCS

The following information covers power requirements for the Model 4200-SCS power connections, power-up characteristics, and warm-up requirements.

Line power

The Model 4200-SCS operates from a line voltage in the range of 100 V to 240 V at a frequency of 50Hz or 60Hz. Line voltage is automatically sensed, but line frequency is not (see Line frequency setting). Check to ensure the operating voltage in your area is compatible.

CAUTION **Operating the instrument on an incorrect line voltage may cause damage, possibly voiding the warranty.**

NOTE *To avoid possible problems caused by electrical transients or line voltage fluctuations, the Model 4200-SCS should be operated from a dedicated power source.*

Line power connection

Perform the following steps to connect the unit to line power and turn it on:

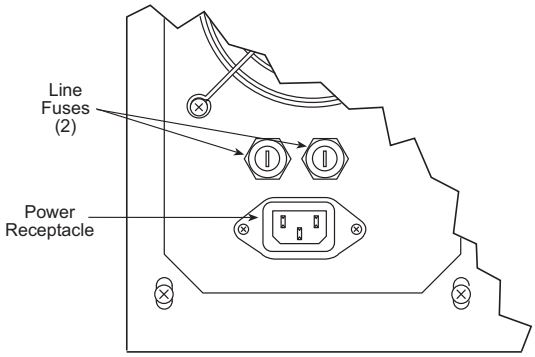
1. Before plugging in the power cord, make sure the front panel power switch is in the off position.
2. Connect the female end of the supplied power cord to the AC receptacle on the rear panel (Figure 2-13).

WARNING ***The large diameter line cord (supplied) must be used to power the Model 4200-SCS. DO NOT use a different line cord.***

3. Connect the other end of the supplied line cord to a grounded AC line power receptacle.

WARNING ***The power cord supplied with the unit contains a separate ground for use with grounded outlets. When proper connections are made, the instrument chassis is connected to power line ground through the ground wire in the power cord. Failure to use a grounded outlet may result in personal injury or death due to electric shock.***

Figure 2-13
Line power receptacle and line fuses location



Line frequency setting

The Model 4200-SCS can be operated either from 50Hz or 60Hz power line sources, but it does not automatically sense the power line frequency at power-up. You can change the line frequency setting using the KCON utility. See Keithley CONfiguration Utility (KCON) in Section 7.

NOTE *Operating the Model 4200-SCS with the wrong line frequency setting may result in noisy readings because the line frequency setting affects SMU line frequency noise rejection.*

Line fuses

Rear panel fuses protect the power line input of the unit. If the line fuses need to be replaced, perform the following steps:

WARNING ***Turn off the power and disconnect the line cord before replacing the fuses.***

1. The fuses are located in two fuse holders above the AC receptacle (Figure 2-13).
2. Using a small slotted screwdriver to remove each fuse holder, push the fuses in and rotate them counterclockwise to remove.
3. Remove the fuses from the fuse holders and replace them with the following type: 250 V, 15 A, 5 × 20 mm, SLOWBLOW.

CAUTION **For continued protection against fire or instrument damage, replace the fuses only with the type and rating shown above. If the instrument repeatedly blows fuses, locate and correct the cause of the problem before replacing the fuses.**

Power-up sequence

During the power-up sequence, the embedded computer starts the basic input/output system (BIOS), and then starts the Microsoft® Windows® operating system (OS). When the OS is running, KITE automatically starts.

Warm-up period

The Model 4200-SCS can be used immediately after being turned on. However, the unit should be allowed to warm up for at least 30 minutes to achieve rated measurement accuracy.

Section 3

Source-Measure Hardware

In this section:

Introduction

This section provides detailed information about the various Model 4200-SCS hardware components, and is arranged as follows:

- **Models 4200-SMU and 4210-SMU overview, page 3-3**: Discusses Models 4200-SMU and 4210-SMU basic source and measure characteristics, basic circuit configurations, operating boundaries, and connectors.
- **Source measure unit (SMU) with Model 4200-PA overview, page 3-14:** Details how the Model
  4200-PA extends Models 4200-SMU and 4210-SMU dynamic range, and covers source and measure characteristics, basic circuit configurations, operating boundaries, connectors, and mounting methods.
- **Ground unit (GNDU) overview, page 3-25:** Provides basic information about using the ground unit, including basic characteristics and connectors.

NOTE *Details for the Keithley pulse cards and scope card (Models 4200-SCP2 or 4200-SCP2HR) are provided in the Reference Manual, Section 11 "Pulse Source-Measure Concepts." Section 16 provides details about the Models 4220-PGU and 4225-PMU pulse cards. It also documents the Model 4225-RPM.*

Models 4200-SMU and 4210-SMU overview

The following paragraphs discuss these aspects of both the Models 4200-SMU and 4210-SMU:

- Basic characteristics
- Basic SMU configuration
- Compliance limit
- Operating boundaries
- Connectors

Basic characteristics

Current characteristics

Current characteristics for both SMUs are summarized in Table 3-1.

Table 3-1
Models 4200-SMU and 4210-SMU current characteristics

Function	4200-SMU	4210-SMU
Current source ranges (full scale/set resolution)	105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50nA 10.5 mA / 500nA 105 mA / 5 μA -	105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50nA 10.5 mA / 500nA 105 mA / 5 μA 1.05 A / 50 μA

Table 3-1
Models 4200-SMU and 4210-SMU current characteristics

Function	4200-SMU	4210-SMU
Current measurement ranges (full scale/nominal resolution)	105 nA / 1pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA / 10 nA 10.5 mA / 100 nA 105 mA / 1 μA -	105 nA / 1 pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA / 10 nA 10.5 mA / 100 nA 105 mA / 1 μA 1.05 A / 10 μA

Voltage characteristics

Table 3-2 summarizes SMU voltage characteristics.

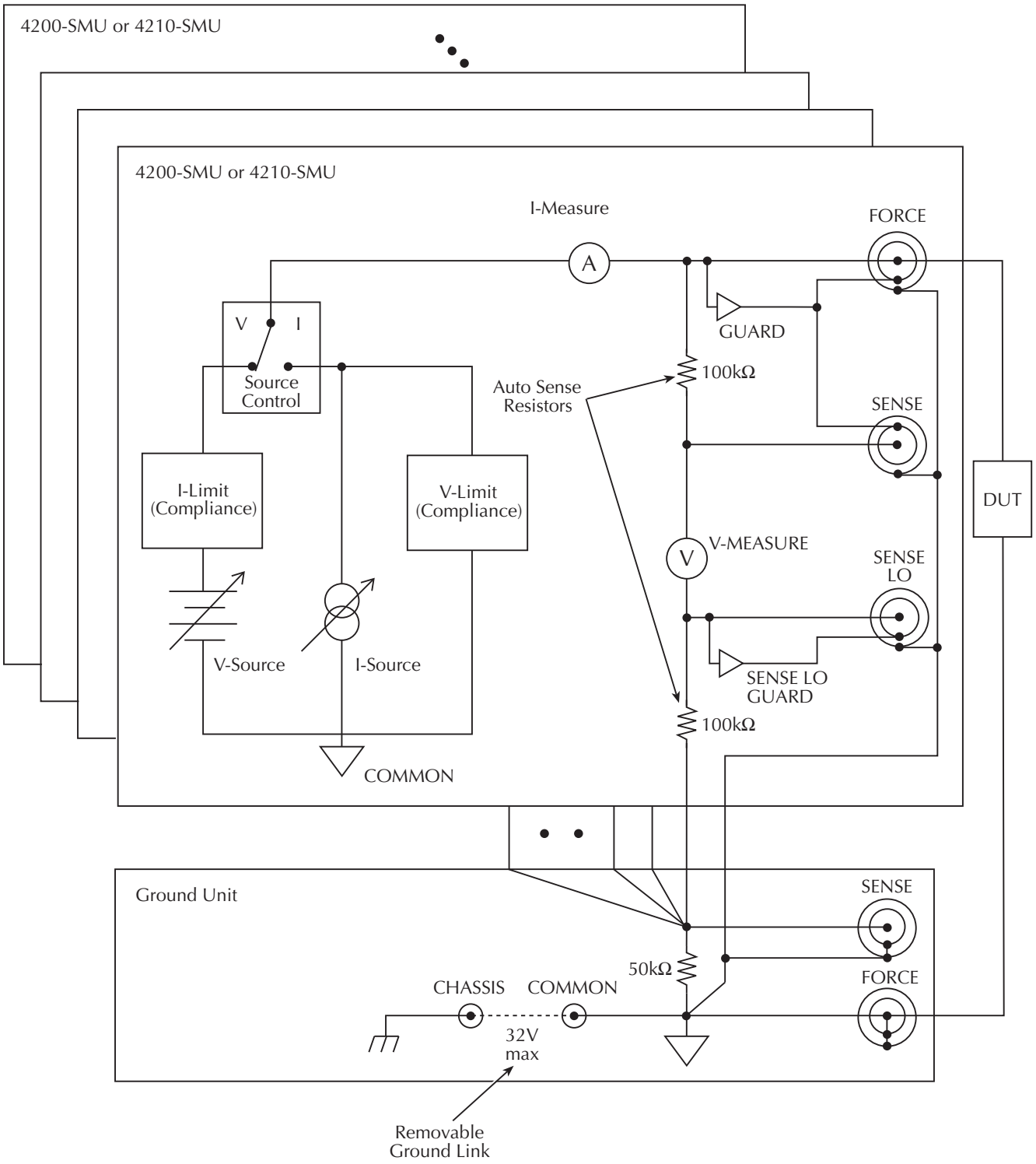
Table 3-2
Models 4200-SMU and 4210-SMU voltage characteristics

Function	4200-SMU	4210-SMU
Voltage source ranges (full scale/set resolution)	210 mV / 5 μV 2.1 V / 50 μV 21V / 500 μV 210 V / 5 mV	210 mV / 5 μV 2.1 V / 50 μV 21 V / 500 μV 210 V / 5 mV
Voltage measurement ranges (full scale/nominal resolution)	210 mV / 1μV 2.1 V / 10 μV 21 V / 100 μV 210 V / 1 mV	210 mV / 1μV 2.1 V / 10 μV 21 V / 100 μV 210 V / 1 mV

Basic SMU circuit configuration

The basic SMU circuit configuration is shown in Figure 3-1. The SMU is essentially a voltage or current source (depending on source function) in series with an I-Meter, and connected in parallel with a V-Meter. The voltage limit (V-limit) and current limit (I-limit) circuits limit the voltage or current to the programmed compliance value. In this local sensing example, the SMU FORCE terminal is connected to DUT HI, while the DUT LO is connected to COMMON. See Basic source-measure connections, page 4-3 in Section 4 and Source-measure configurations, page 5-12 for more detailed information.

Figure 3-1
Basic SMU source-measure configuration



Compliance limit

When sourcing voltage, the Models 4200-SMU and 4210-SMU can be programmed to limit current (I-limit). Conversely, when sourcing current, the SMUs can be programmed to limit voltage (V-limit). The SMU will limit the output to the programmed compliance value regardless of load.

Types of compliance

There are two types of compliance: real and range. Depending on which value is lower, the output will clamp at either the programmed compliance setting (real compliance) or at the maximum possible compliance value for the fixed measurement range (range compliance). This clamping action effectively limits the power that can be delivered to the device. When the SMU is acting as a current source, the voltage will limit at the compliance value; conversely, the current will limit at the compliance value when the SMU is acting as a voltage source. Note that range compliance cannot occur if the SMU measurement circuit is configured for autorange (however, range compliance can momentarily occur while the unit is up-ranging range). To avoid range compliance, use autorange.

When in range compliance, the source output will limit at the maximum compliance value for the fixed measurement range (not the programmed compliance value). For example, if compliance is set to 1V and the measurement range is 200 mV, output voltage will clamp at 210 mV.

When in real compliance, the source will limit at the programmed compliance value. For example, if the programmed compliance voltage is set to 1 V and the measurement range is 2V, output voltage will clamp at 1 V.

Maximum and minimum compliance values

Table 3-3 summarizes maximum and minimum current compliance limits according to range, while Table 3-4 lists voltage compliance limits.

Table 3-3
Models 4200-SMU and 4210-SMU current compliance limits

Measure range	Maximum compliance value	Minimum compliance value
100 nA	±105 nA	±10 nA
1 μA	±1.05 μA	±100 nA
10 μA	±10.5 μA	±1 μA
100 μA	±105 μA	±10 μA
1 mA	±1.05 mA	±100 μA
10 mA	±10.5 mA	±1 mA
100 mA	±105 mA	±10 mA
1 A*	±1.05 A	±100 mA

*Model 4210-SMU only.

Table 3-4
Models 4200-SMU and 4210-SMU voltage compliance limits

Measure range	Maximum compliance value	Minimum compliance value
200 mV	±210 mV	±20 mV
2 V	±2.1 V	±200 mV
20 V	±21 V	±2 V
200 V	±210 V	±20 V

Using minimum compliance

The minimum compliance value is particularly applicable when measurement autorange is disabled. When measurement autorange is disabled, the compliance value cannot be set below the minimum value specified in Table 3-3 and Table 3-4. When autorange is enabled, the programmed compliance value cannot be set below 10nA when sourcing voltage, or below 20 mV when sourcing current.

Operating boundaries

Source or sink

Depending on how they are programmed and what is connected to the output (load or source), the SMUs can operate in any of the four quadrants. The four quadrants of operation for the Models 4200-SMU and 4210-SMU are shown in Figure 3-2 and Figure 3-3, respectively. When operating in the first (I) or third (III) quadrant, the SMUs are operating as a source (V and I have the same polarity). As a source, the SMUs are delivering power to a load. When operating in the second (II) or fourth (IV) quadrant, the SMUs are operating as a sink (V and I have opposite polarity). As a sink, they are dissipating power rather than sourcing it.

Model 4200-SMU: In the general operating boundaries in Figure 3-2, the 100 mA, 20 V and 10 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the Model 4200-SMU are 105 mA, 21 V and 10.5 mA, 210 V. Also note that the boundaries are not drawn to scale.

Model 4210-SMU: In Figure 3-3, the 1A, 20 V and 100 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the Model 4210-SMU are 1.05 A, 21 V and 105 mA, 210 V. Again, the boundaries are not drawn to scale.

Figure 3-2
Model 4200-SMU operating boundaries

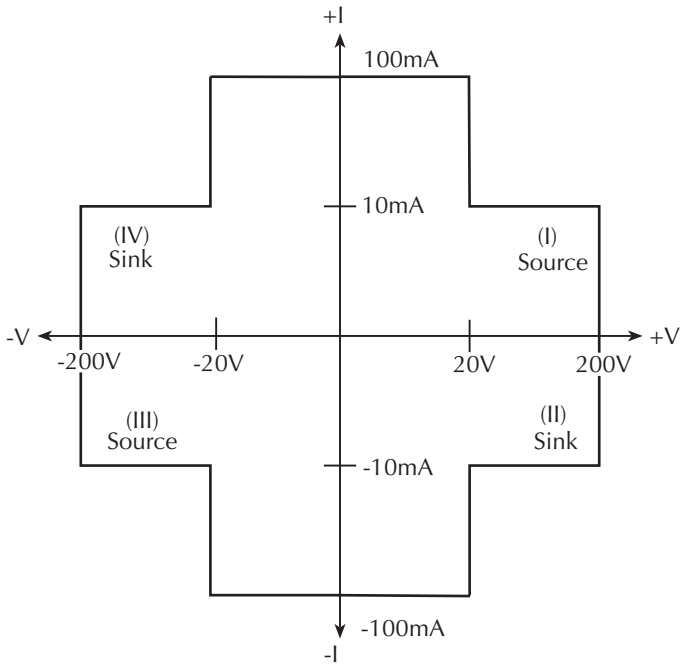
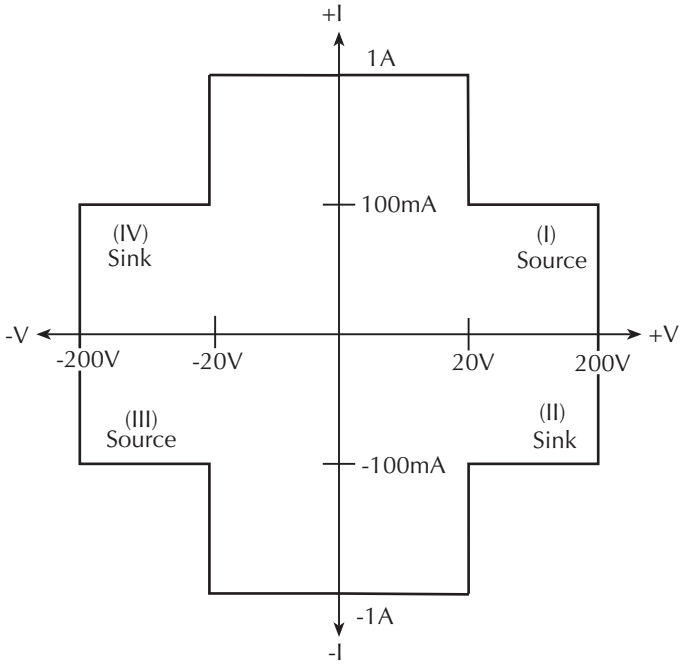


Figure 3-3
Model 4210-SMU operating boundaries



I-Source operating boundaries

Limit lines are boundaries that represent the operating limits of the SMU for a certain quadrant of operation. The operating point can be anywhere inside (or on) these limit lines. The limit line boundaries for the other quadrants are similar.

Figure 3-4 and Figure 3-5 show the operating boundaries for the I-Source. Only the first quadrant of operation is covered; operation in the other three quadrants is similar.

Model 4200-SMU: As shown in Figure 3-4A, the Model 4200-SMU can output up to 105 mA at 21 V, or 10.5 mA at 210 V.

Model 4210-SMU: As shown in Figure 3-4B, the Model 4210-SMU can output up to 1.05 A at 21 V, or 105 mA at 210 V.

Figure 3-5 shows the limit lines for the I-Source. The current source limit line represents the maximum source value possible for the selected current source range. For example, the current source limit line is at 105 mA on the 100 mA current source range. The voltage compliance limit line represents the actual compliance that is in effect.

Figure 3-4
Models 4200-SMU and 4210-SMU I-Source output characteristics

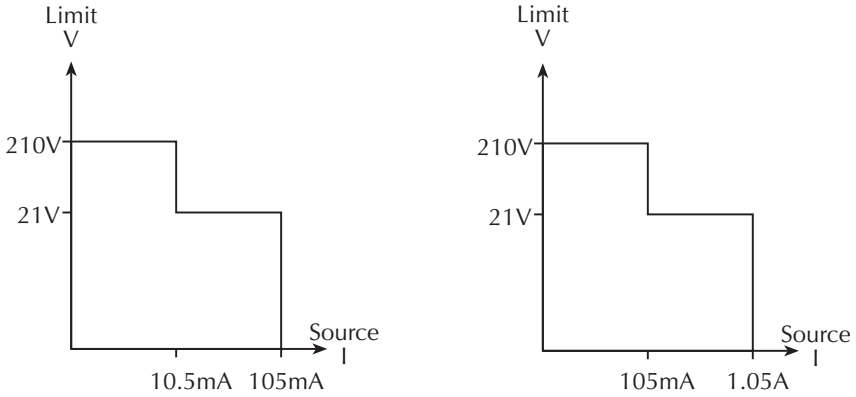
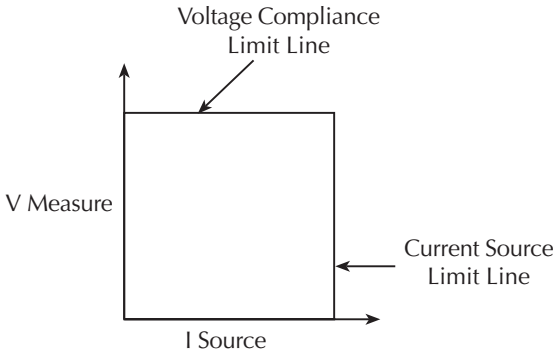


Figure 3-5
Models 4200-SMU and 4210-SMU I-Source limit lines



I-Source operation examples

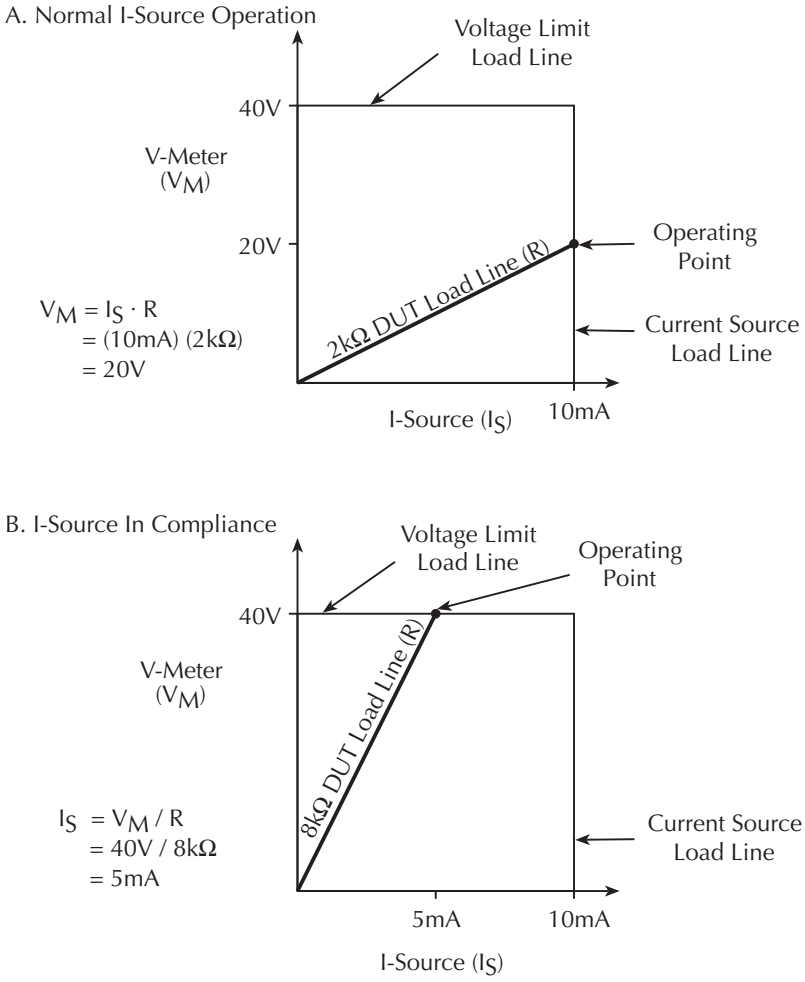
Figure 3-6 shows operation examples for resistive loads that are 2k Ω and 8k Ω, respectively. For these examples, the SMU is programmed to source 10 mA and limit (compliance) 40 V. In Figure 3-6A, the SMU is sourcing 10 mA to the 2k Ω load, and subsequently measures 20 V. As shown, the load line for 2k Ω intersects the 10 mA current source line at 20 V, which is below the programmed voltage limit.

Figure 3-6B shows what happens if the resistance of the load is increased to 8k Ω. The DUT load line for 8k Ω intersects the 40 V voltage compliance limit line, placing the SMU in compliance. In compliance, the SMU will not be able to source its programmed current (10 mA). For the 8k Ω DUT, the SMU will only output 5 mA (at the 40 V limit).

Notice that as resistance increases, the slope of the DUT load line increases. As resistance approaches infinity (open output), the SMU will source virtually 0 mA at 40 V. Conversely, as resistance decreases, the slope of the DUT load line decreases. At zero resistance (shorted output), the SMU will source 10 mA at virtually 0 V.

Regardless of the load, voltage will never exceed the programmed compliance of 40 V.

Figure 3-6
I-Source operating examples



V-Source operating boundaries

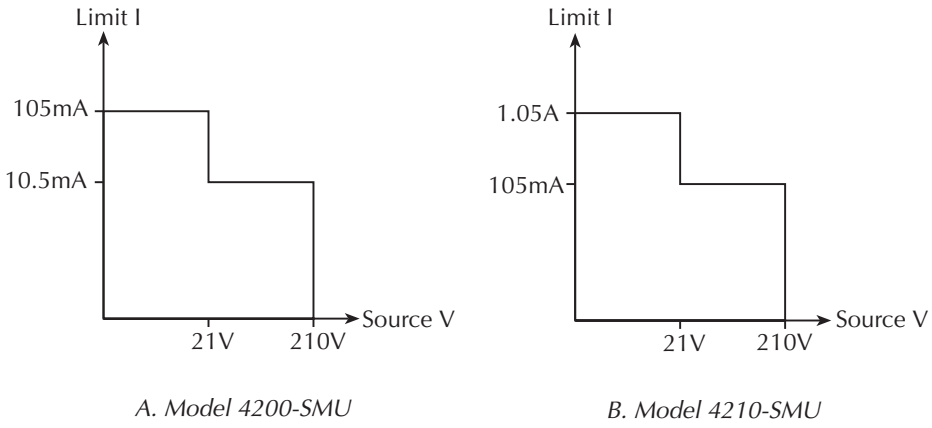
Figure 3-7 and Figure 3-8 show the operating boundaries for the V-Source. Only the first quadrant of operation is covered; operation in the other three quadrants is similar.

Model 4200-SMU: As shown in Figure 3-7A, the Model 4200-SMU can output up to 21 V at 105 mA, or 210 V at 10.5 mA.

Model 4210-SMU: As shown in Figure 3-7B, the Model 4210-SMU can output up to 21 V at 1.05 A, or 210 V at 105 mA.

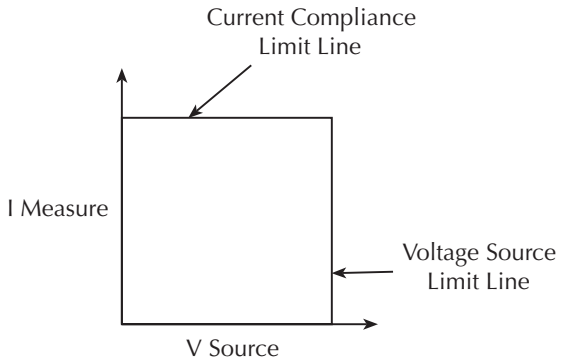
Figure 3-8 shows the limit lines for the V-Source. The voltage source limit line represents the maximum source value possible for the selected voltage source range. For example, the voltage source limit line is at 21 V for the 20 V source range. The current compliance limit line represents the actual compliance in effect. These limit lines are boundaries that represent the operating limits of the SMU for this quadrant of operation. The operating point can be anywhere inside (or on) these limit lines. The limit line boundaries for the other quadrants are similar.

Figure 3-7
Models 4200-SMU and 4210-SMU V-Source output characteristics



A. Model 4200-SMU *B. Model 4210-SMU*

Figure 3-8
Models 4200-SMU and 4210-SMU V-Source limit lines



V-Source operation examples

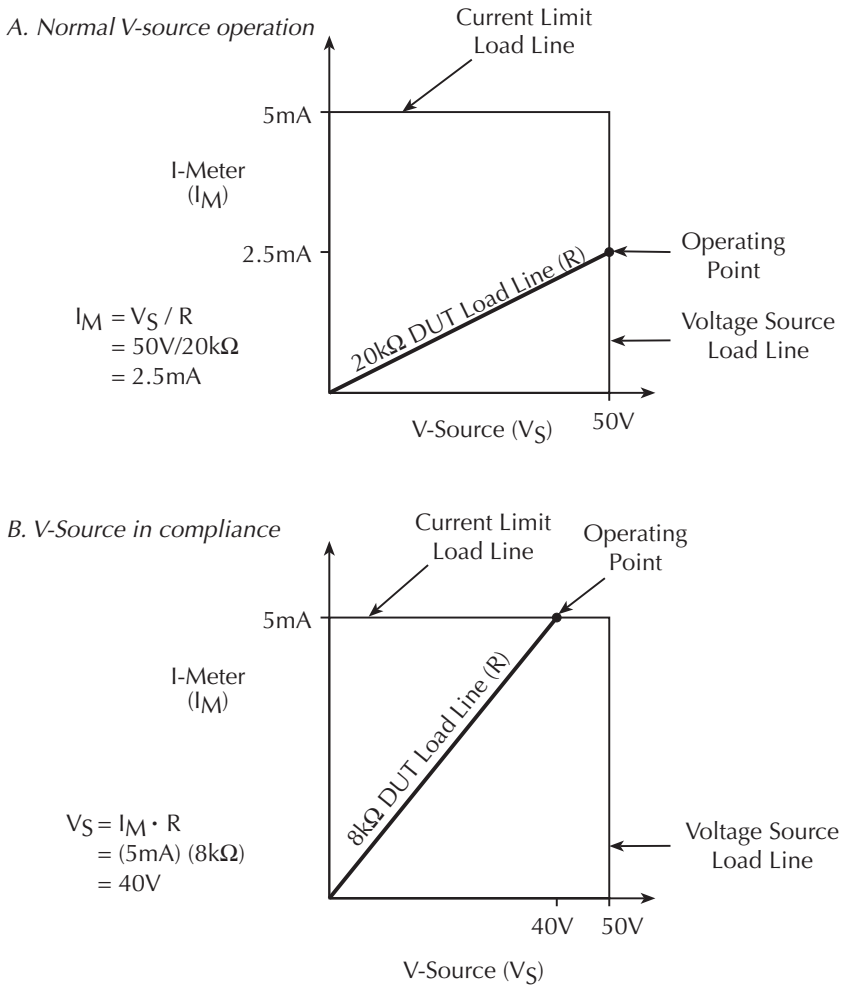
Figure 3-9 shows operation examples for resistive loads that are 20 k Ω and 8k Ω, respectively. For these examples, the SMU is programmed to source 50 V and limit 5 mA. In Figure 3-9A, the SMU is sourcing 50 V to the 20 k Ω load and subsequently measures 2.5 mA, which is well within the 5 mA programmed current limit. As shown, the load line for 20 k Ω intersects the 50 V voltage source line at 2.5 mA.

Figure 3-9B shows what happens if the resistance of the load is decreased to 8k Ω. The DUT load line for 8k Ω intersects the current compliance limit line placing the SMU in compliance. In compliance, the SMU will not be able to source its programmed voltage (50 V). For the 8k Ω DUT, the SMU will only output 40 V (at the 5 mA limit).

Notice that as resistance decreases, the slope of the DUT load line increases. As resistance approaches infinity (open output), the SMU will source virtually 50 V at 0 mA. Conversely, as resistance decreases, the slope of the DUT load line increases. At zero resistance (shorted output), the SMU will source virtually 0 V at 5 mA.

Regardless of the load, current will never exceed the programmed compliance of 5 mA.

Figure 3-9
V-Source operating examples



Source I measure I and source V measure V

The SMU can measure the function it is sourcing. When sourcing a voltage, you can also measure voltage. Conversely, if you are sourcing current, you can also measure the output current. For these measure source operations, the measure range is always the same as the source range.

This feature is valuable when operating with the source in compliance, or when additional overall accuracy is desired. When in compliance, the programmed source value is not reached. Thus, measuring the source lets you measure the actual output voltage.

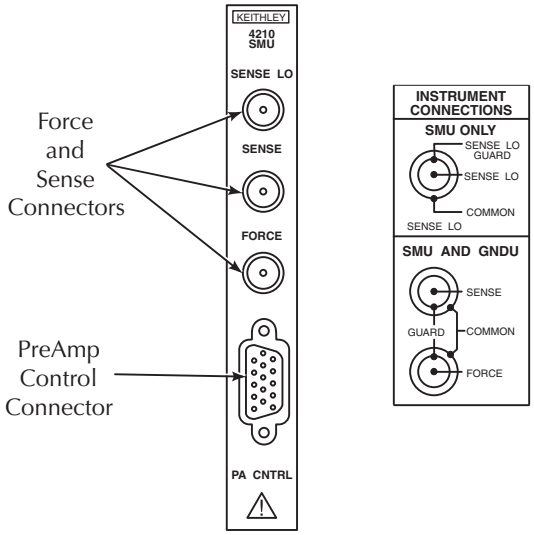
SMU terminals and connectors

The locations and configuration of the Models 4200-SMU and 4210-SMU terminals are shown in Figure 3-10. Basic information about these terminals is summarized below. Refer to the Basic source-measure connections, page 4-3 for additional information regarding SMU signal connections.

WARNING ***Asserting the interlock will allow the SMU and PreAmp terminals to become hazardous, exposing the user to possible electrical shock that could result in personal injury or death. SMU and PreAmp terminals should be considered hazardous even if the outputs are programmed to be low voltage. Precautions must be taken to prevent a shock hazard by surrounding the test device and any unprotected leads (wiring) with double insulation for 250 volts, Category I. exposing the user to electrical shock that could result in personal injury or death.***

CAUTION **The maximum allowed voltage between COMMON and chassis ground is ±32V DC.**

Figure 3-10
Models 4200-SMU and 4210-SMU connectors



FORCE terminal

The FORCE terminal is a miniature triaxial connector used to apply the SMU FORCE signal to the DUT when a preamp is not being used. Note that the center pin is FORCE, the inner shield is GUARD, and the outer shield is circuit COMMON.

SENSE terminal

The SENSE terminal is a miniature triaxial connector used to apply the SMU SENSE signal to the DUT in a remote sense application when the preamp is not being used. The center pin is SENSE, the inner shield is GUARD, and the outer shield is circuit COMMON. Nominal internal auto-sense resistance appears between SENSE and FORCE.

NOTE *The SENSE terminal does not need to be connected to the DUT for the SMU to operate correctly. Remote sensing is automatic. If SENSE is connected to the DUT, errors due to voltage drops in the FORCE path between the SMU and the DUT will be eliminated; otherwise, the SMU will sense locally.*

SENSE LO terminal

The SENSE LO terminal is a miniature triaxial connector used to apply the SMU SENSE LO signal to the DUT in a full-kelvin remote sense application. The center pin is SENSE LO, the inner shield is SENSE GUARD, and the outer shield is circuit COMMON. Nominal internal auto-sense resistance appears between SENSE LO GUARD and COMMON.

NOTE *Generally the remote sense capability of the ground unit should be used instead of the SENSE LO of an SMU. If it is necessary to use the SENSE LO of an SMU, the SENSE LO terminals of all SMUs being used in that Model 4200-SCS should be connected to the DUT.*

PA CNTRL connector

The PA CNTRL (Preamp Control) terminal is a 15-pin D connector that provides both power and signal connections to the Model 4200-PA Remote preamp. Preamp can either be mounted and connected directly to the SMU, or it can be connected to the SMU via a cable (Model 4200-RPC-X) when mounted remotely. Refer to Model Source measure unit (SMU) with Model 4200-PA overview below for more information on the preamp.

Source measure unit (SMU) with Model 4200-PA overview

The following paragraphs discuss these aspects of the Model 4200-PA remote preamp:

- Basic characteristics
- Basic circuit configuration
- Compliance limit
- Operating boundaries
- Connectors
- Preamp mounting

Basic characteristics

Current characteristics

Current characteristics of the Model 4200-SMU and 4210-SMU when used with the Model 4200-PA are summarized in Table 3-5. Note that the preamp extends the current source-measure dynamic range of the Model 4200-SMU and 4210-SMU downward by five decades. The lowest current range available *without* the preamp is 100nA full scale, while the lowest range *with* the preamp is 1pA full scale.

Table 3-5
SMU with Model 4200-PA current characteristics

Function	4200-SMU with 4200-PA	4210-SMU with 4200-PA
Current source ranges (full scale/set resolution)	1.05 pA / 50 aA 10.5 pA / 500 aA 100.5 pA / 5 fA 1.05 nA / 50 fA 10.5 nA / 500 fA 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA -	1.05 pA / 50 aA 10.5 pA / 500 aA 100.5 pA / 5 fA 1.05 nA / 50 fA 10.5 nA / 500 fA 105 nA / 5 pA 1.05 μA / 50 pA 10.5 μA / 500 pA 105 μA / 5 nA 1.05 mA / 50 nA 10.5 mA / 500 nA 105 mA / 5 μA 1.05 A / 50 μA
Current measurement ranges (full scale/nominal resolution)	1.05 pA / 10 aA 10.5 pA / 100 aA 100.5 pA / 1 fA 1.05 nA / 10 fA 10.5 nA / 100 fA 105 nA / 1 pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA / 10 nA 10.5 mA / 100 nA 105 mA / 1 μA -	1.05 pA / 10 aA 10.5 pA / 100 aA 100.5 pA / 1 fA 1.05 nA / 10 fA 10.5 nA / 100 fA 105 nA / 1pA 1.05 μA / 10 pA 10.5 μA / 100 pA 105 μA / 1 nA 1.05 mA / 10 nA 10.5 mA / 100 nA 105 mA / 1 μA 1.05 A / 10 μA

Voltage characteristics

Table 3-6 summarizes a SMU with Model 4200-PA voltage characteristics that are identical to those for the SMUs alone.

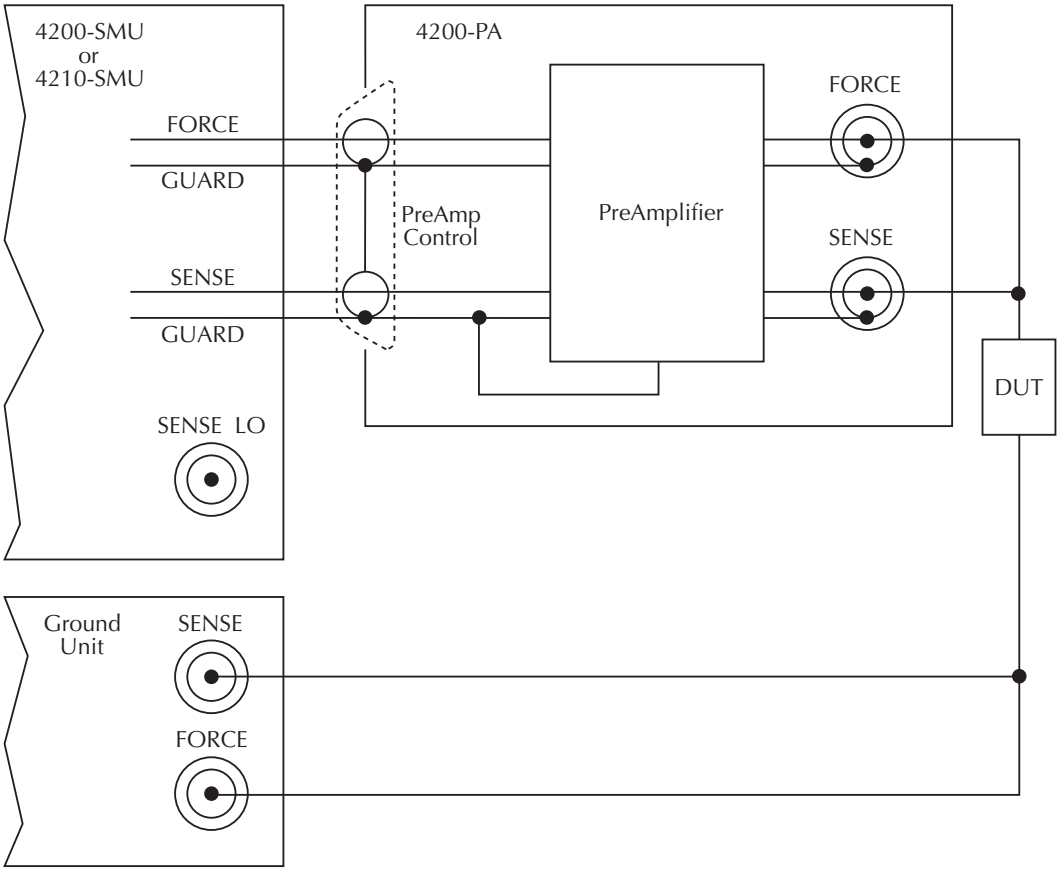
Table 3-6
SMU with Model 4200-PA voltage characteristics

Function	4200-SMU with 4200-PA	4210-SMU with 4200-PA
Voltage source range (full scale/set resolution)	210 mV / 5μV 2.1 V / 50μV 21 V / 500μV 210 V / 5 mV	210 mV / 5μV 2.1 V / 50μV 21 V / 500μV 210 V / 5 mV
Voltage measurement range (full scale/nominal resolution)	210 mV / 1 μV 2.1 V / 10 μV 21 V / 100 μV 210 V / 1 mV	210 mV / 1 μV 2.1 V / 10 μV 21 V / 100 μV 210 V / 1 mV

Basic SMU/PreAmp circuit configuration

Basic SMU/Preamp circuit configuration is shown in Figure 3-11. This configuration is similar to the SMU configuration discussed earlier, with the exception of the preamp, which adds low-current source-measure capabilities. Preamp FORCE terminal is connected to DUT HI, while DUT LO is connected to COMMON. See Basic source-measure connections, page 4-3, and Source-measure configurations, page 5-12, for more detailed information.

Figure 3-11
Basic SMU/PreAmp source-measure configuration



Compliance limit

A current limit can be programmed for a SMU with a Model 4200-PA when it is sourcing voltage. Conversely, a voltage limit can be programmed when sourcing current. The compliance limit characteristics are the same for a SMU with a Model 4200-PA as for a SMU alone. See Compliance limit, page 3-6 earlier in this section for more detailed information on types of compliance.

Maximum and minimum compliance values

Table 3-7 summarizes current compliance limits for the preamp according to range, while Table 3-8 lists SMU preamp voltage compliance limits.

Table 3-7
SMU with Model 4200-PA current compliance limits

Measure Range	Maximum Compliance Value	Minimum Compliance Value
1pA	±1.05 pA	±100 fA
10 pA	±10.5 pA	±1 pA
100 pA	±105 pA	±10 pA
1 nA	±1.05 nA	±100 pA
10 nA	±10.5 nA	±1 nA
100 nA	±105 nA	±10 nA
1 µA	±1.05 µA	±100 nA

Table 3-7
SMU with Model 4200-PA current compliance limits

Measure Range	Maximum Compliance Value	Minimum Compliance Value
10 μA	±10.5 μA	±1 μA
100μA	±105μA	±10μA
1mA	±1.05 mA	±100μA
10 mA	±10.5 mA	±1mA
100 mA	±105 mA	±10 mA
1A*	±1.05 A	±100 mA

*1A range only with Model 4210-SMU.

Table 3-8
SMU with Model 4200-PA voltage compliance limits

Measure Range	Maximum Compliance Value	Minimum Compliance Value
200 mV	±210 mV	±20 mV
2V	±2.1V	±200 mV
20 V	±21V	±2V
200 V	±210 V	±20 V

Using minimum compliance

The minimum compliance value is particularly applicable when measurement autorange is disabled. When measurement autorange is disabled, the compliance value cannot be set below the minimum value specified in Table 3-7 and Table 3-8. When autorange is enabled, the programmed compliance value cannot be set below 100fA when sourcing voltage, or below 20 mV when sourcing current.

Operating boundaries

As with the SMUs alone, adding Model 4200-PA preamp also allows operation in any of the four quadrants. The four quadrants of operation for the Model 4200-PA with the Models 4200-SMU and 4210-SMU are shown in Figure 3-12 and Figure 3-13 respectively. For a more detailed discussion on V-Source and I-Source operating boundaries, see Operating boundaries, page 3-7 in the Models
4200-SMU and 4210-SMU overview earlier in this section.

Model 4200-SMU with Model 4200-PA: In the general operating boundaries in Figure 3-12, the 100 mA, 20 V and 10 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the Model 4200-SMU/4200-PA are 105 mA, 21V and 10.5 mA, 210 V. Also note that the boundaries are not drawn to scale.

Model 4210-SMU with Model 4200-PA: In Figure 3-13, the 1A, 20 V and 100 mA, 200 V magnitudes are nominal values. The actual maximum output magnitudes of the Model 4210-SMU/4200-PA are 1.05 A, 21V and 105 mA, 210 V. Again, the boundaries are not drawn to scale.

Figure 3-12
Model 4200-SMU/4200-PA operating boundaries

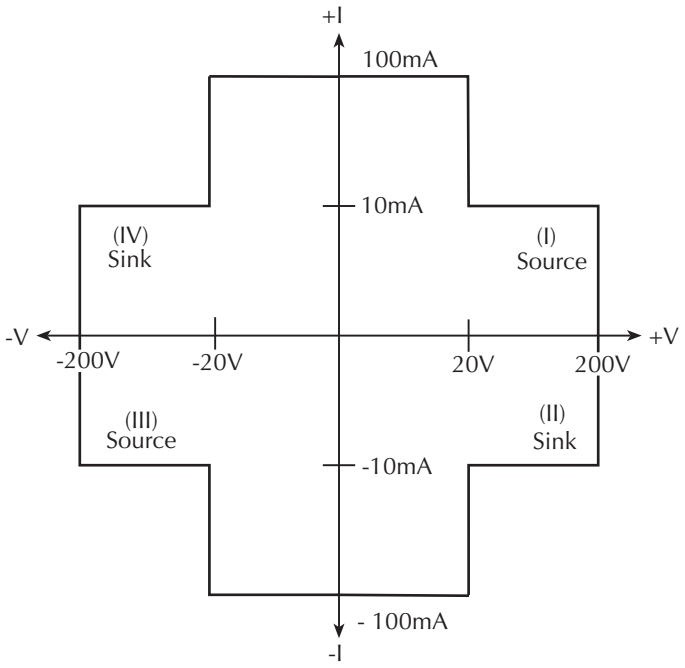
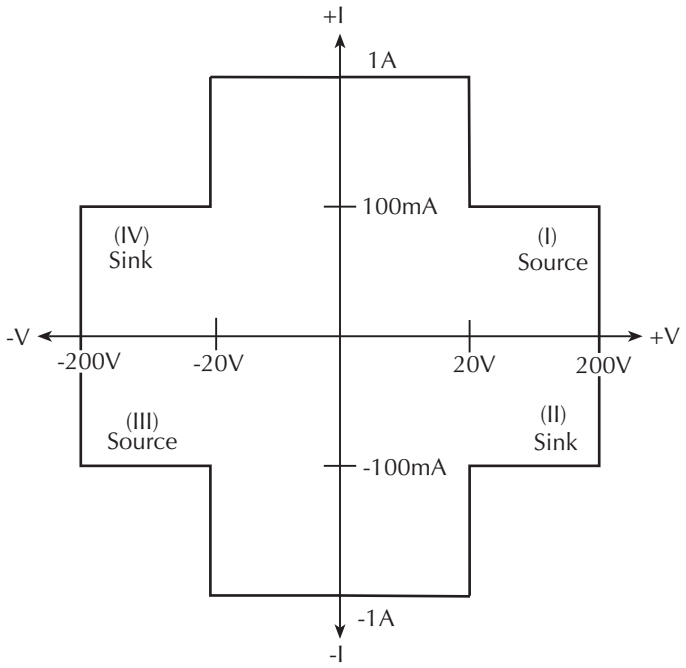


Figure 3-13
Model 4210-SMU/4200-PA operating boundaries



PreAmp terminals and connectors

The locations and configuration of the Model 4200-PA terminals are shown in Figure 3-14. Basic information about these terminals is summarized below. Refer to the Basic source-measure connections, page 4-3 for additional information regarding making preamp signal connections.

WARNING ***The PreAmp terminals can carry hazardous voltage if the safety interlock is asserted, exposing the user to possible electrical shock that could result in personal injury or death. See Control and data connections, page 4-21 in Section 4 for additional information regarding safety interlock connections.***

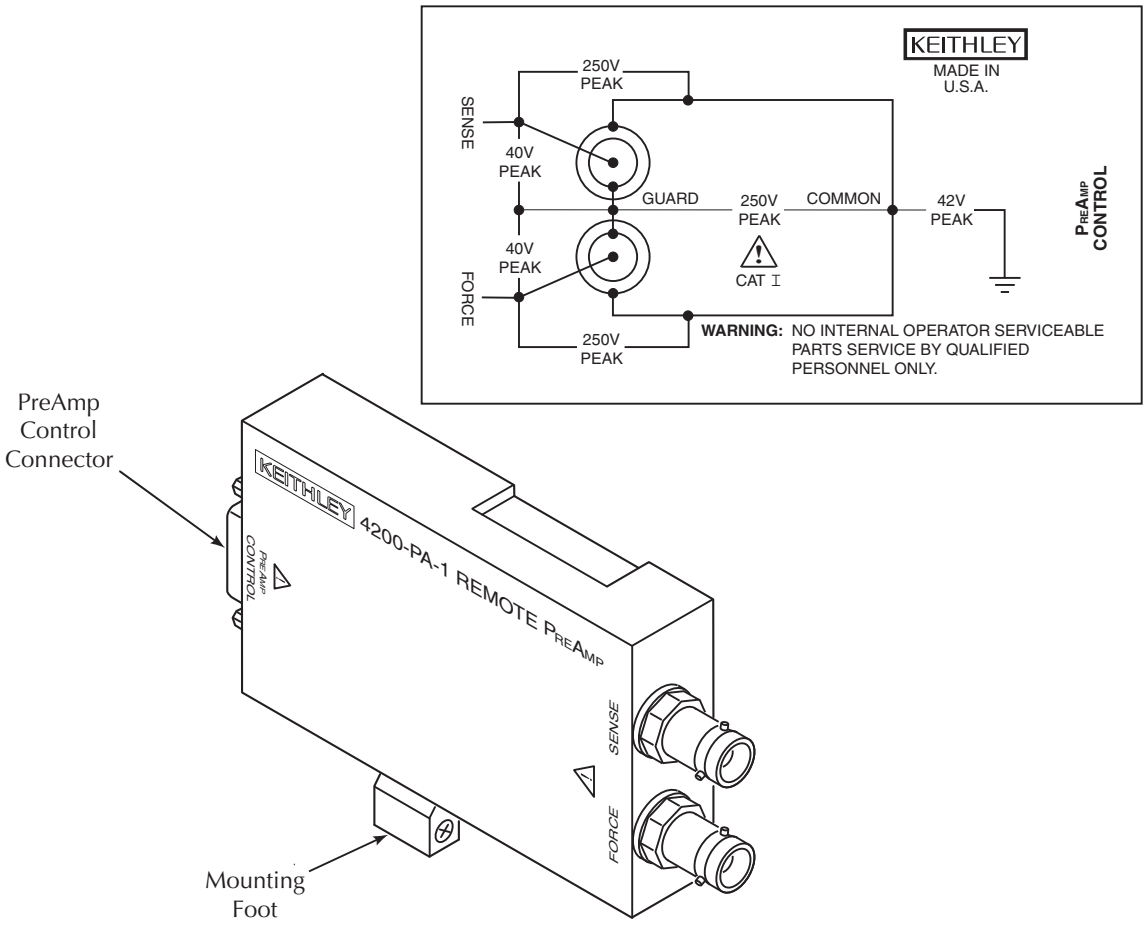
CAUTION **The maximum allowed voltages between the various PreAmp signals are as follows:**

- **COMMON to chassis ground: 32Vpeak**
- **GUARD to COMMON: 250 V peak**
- **SENSE or FORCE to GUARD: 40 V peak**

FORCE terminal

The FORCE terminal is a standard triaxial connector used to apply the preamp FORCE signal to the DUT. Note that the center pin is FORCE, the inner shield is GUARD, and the outer shield is circuit COMMON.

Figure 3-14
Model 4200-PA connectors



SENSE terminal

The SENSE terminal is a standard triaxial connector used to apply the preamp SENSE signal to the DUT in a remote sense application. The center pin is SENSE, the inner shield is GUARD, and the outer shield is circuit COMMON. Nominal internal auto-sense resistance appears between SENSE and FORCE.

NOTE *The SENSE terminal does not need to be connected to the DUT for the PreAmp to operate correctly. Remote sensing is automatic. If SENSE is connected to the DUT, errors due to voltage drops in the FORCE path between the PreAmp and the DUT will be eliminated. Otherwise, the PreAmp will sense locally.*

Preamp CONTROL connector

The preamp CONTROL connector connects to the SMU PA CNTRL connector and provides both power and signal connections from the Models 4200-SMU or 4210-SMU to the Model 4200-PA preamp.

PreAmp mounting

The preamp may either be mounted directly to the Models 4200-SMU or 4210-SMU on the mainframe rear panel, or mounted and connected remotely.

NOTE *As shipped, any Model 4200-PA units ordered with the Model 4200-SCS will be factory-mounted on the rear panel. Do not remove the PreAmps from the mainframe unless they are to be mounted at a remote site.*

NOTE *The PreAmps are matched to the SMUs that they are connected to. If mounting PreAmps at a remote site, make sure each PreAmp is connected to its original SMU.*

WARNING ***The PreAmp terminals can carry hazardous voltage if the safety interlock is asserted, exposing the user to possible electrical shock resulting in personal injury or death. See "Control and data connections, page 4-21" in Section 4 for additional information regarding safety interlock connections.***

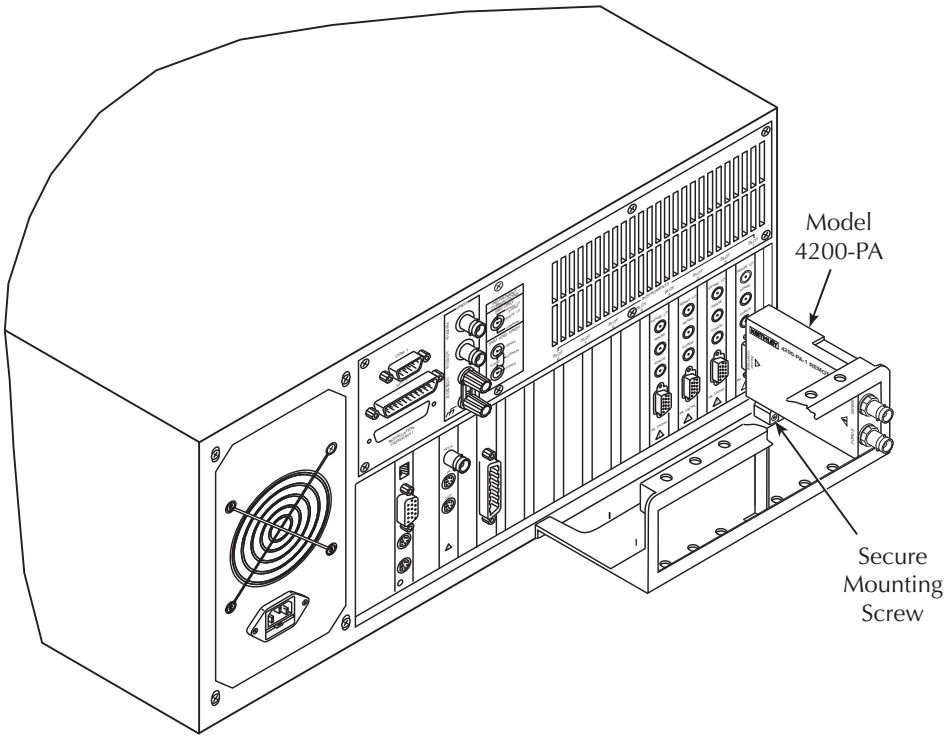
CAUTION **Turn off system power before connecting or disconnecting the PreAmp. Failure to do so may result in SMU or PreAmp damage, possibly voiding the warranty.**

Rear panel mounting

NOTE *Make sure system power is turned off before removing or installing the PreAmps.*

A mounting foot (see Figure 3-14) secures the preamp to the rear panel. Also, a mounting bracket provides extra support for all the preamps as shown in Figure 3-15. If you remove the preamps to mount them at a remote site, ensure that you install the screws in the chassis and retain the bracket for future use.

Figure 3-15
PreAmp rear panel mounting



Remote PreAmp mounting

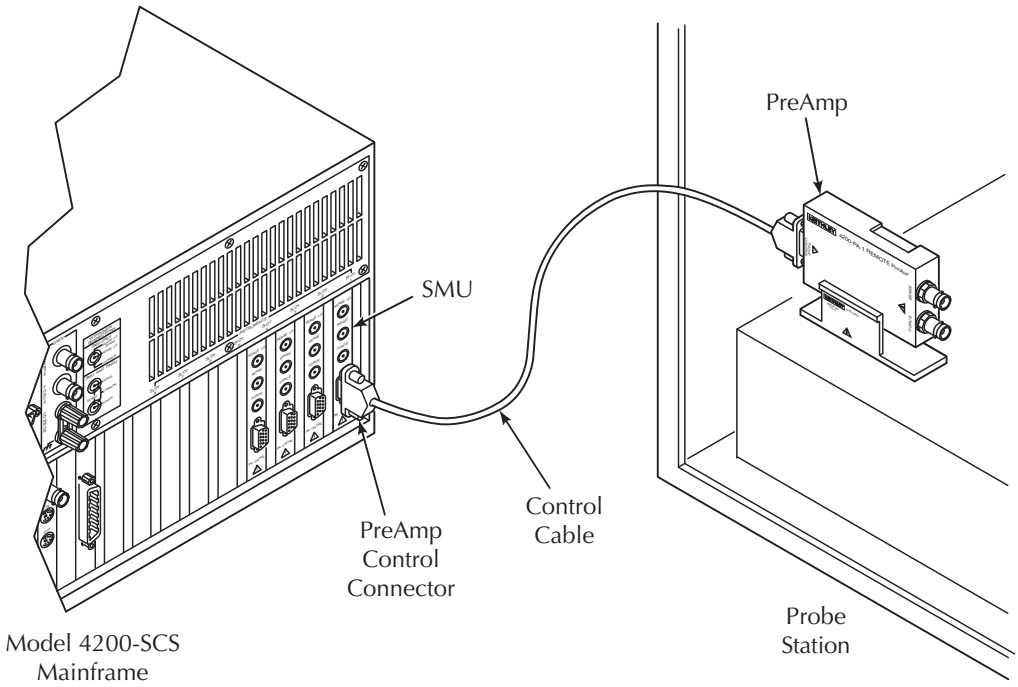
The Model 4200-PA can be mounted remotely using one of the optional mounting kits (See the Options and accessories, page 1-9). Follow the general steps below to remotely mount and connect the preamp. Refer to the instructions provided with the particular remote mounting kit for detailed information on physically mounting the preamp module.

1. Ensure system power is turned off.
2. Mount the preamp at the desired remote location using the appropriate mounting kit.
3. Connect the control/power cable between the preamp control connector on the preamp and the PA CNTRL connector on the SMU as shown in Figure 3-16.

NOTE *The PreAmps are matched to the SMUs that they were originally connected to. Ensure that each PreAmp is connected to its original SMU.*

4. Ensure that the connecting cable is secure at both ends.

Figure 3-16
Typical PreAmp remote mounting



Ground unit (GNDU) overview

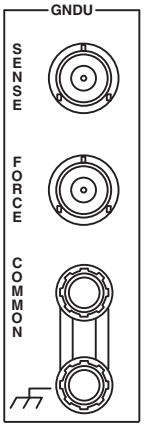
These aspects of the Model 4200-SCS ground unit are covered below:

- Basic characteristics
- Basic circuit configurations
- Connectors

Basic characteristics

The ground unit (see Figure 3-17) provides convenient access to circuit COMMON, which is the measurement ground signal shared by all installed Model 4200-SCS instrumentation. In addition, the GNDU SENSE terminal provides access to the SMU SENSE LO signals.

Figure 3-17
Ground unit



Basic ground unit characteristics are summarized in Table 3-9.

Table 3-9
Basic ground unit characteristics

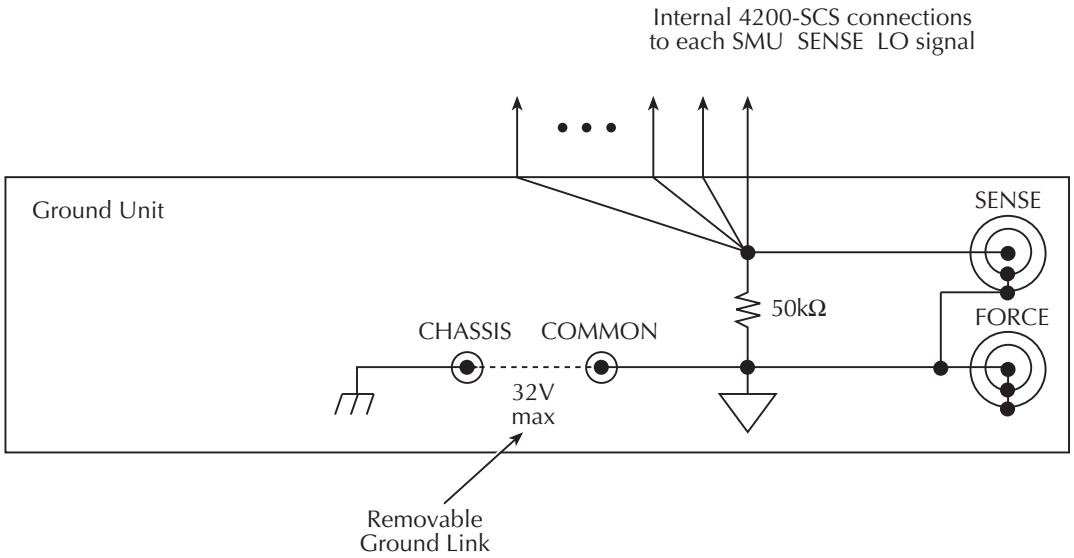
Characteristic	Description
Maximum current (FORCE triax connector)	2.6 A
Maximum current (COMMON binding post connector)	9 A
Maximum FORCE path/cable resistance	1Ω
Maximum SENSE path/cable resistance	10Ω

Basic circuit configurations

Ground unit connections

Figure 3-18 shows how the various GNDU signals are related to the SMU signals. Note that the GNDU FORCE signal is circuit COMMON. The GNDU SENSE terminal is connected to each SMU SENSE LO signal through a unique auto-sense resistor. When the GNDU SENSE signal is connected to a DUT, all measurements will be made relative to this DUT connection.

Figure 3-18
Ground unit connections



This page left blank intentionally.

Ground unit DUT connections

Figure 3-19 shows the connections necessary to use the GNDU in conjunction with a SMU to make full-kelvin remote sense measurements. Similarly, Figure 3-20 includes the preamp. As shown in these figures, the GNDU FORCE signal provides the return path for SMU or preamp FORCE current. See the Basic source-measure connections, page 4-3 for detailed information on the ground unit, SMU, and preamp connections.

Figure 3-19
Full-Kelvin SMU/ground unit connections

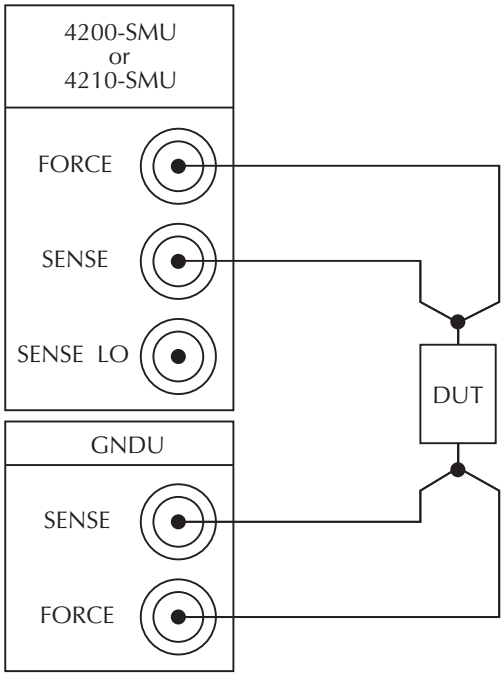
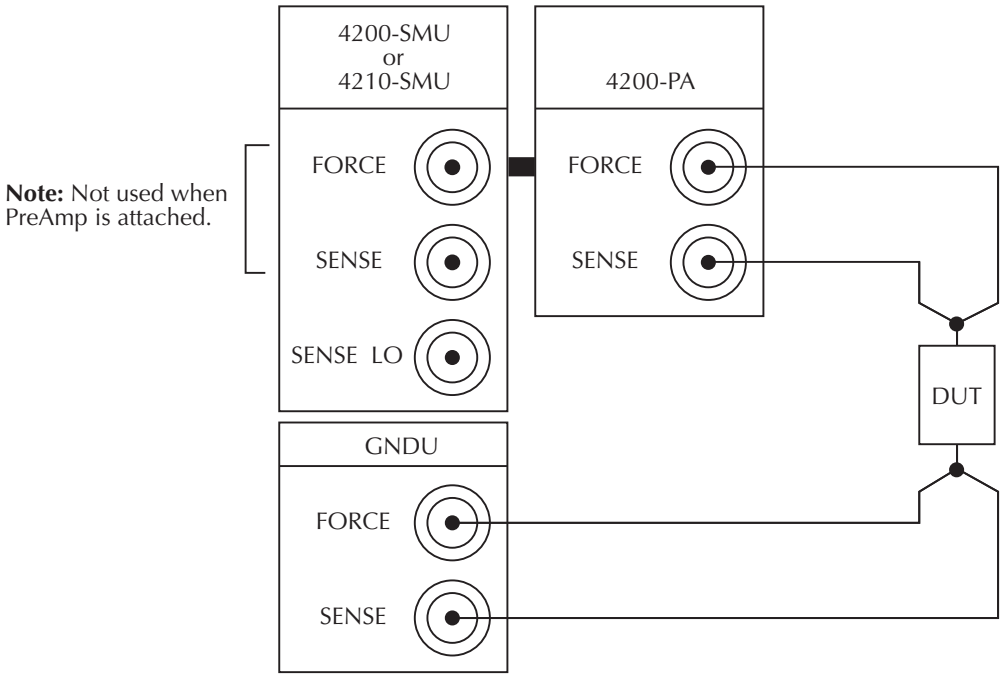


Figure 3-20
Full-Kelvin PreAmp/ground unit connections



Ground unit terminals and connectors

The locations and configuration of the GNDU terminals are shown in Figure 3-17. Basic information about these connectors is summarized below. Refer to the Basic source-measure connections, page 4-3 for additional information regarding ground unit signal connections.

CAUTION **The maximum allowed voltage between circuit COMMON and chassis ground is ±32V DC.**

FORCE terminal

The FORCE terminal is a standard triaxial connector used as a return path for the SMU or preamp FORCE current. The center pin is FORCE, the inner shield is GUARD, and the outer shield is circuit COMMON.

NOTE *The ground unit FORCE and GUARD signal terminals are connected to circuit COMMON.*

SENSE terminal

The SENSE terminal is a standard triaxial connector used to apply the ground unit SENSE signal to the DUT in a remote sense application. The center pin is SENSE, the inner shield is GUARD, and the outer shield is circuit COMMON. When the ground unit SENSE signal is connected to a DUT, all SMU/Preamp measurements will be made relative to this DUT connection.

COMMON terminal

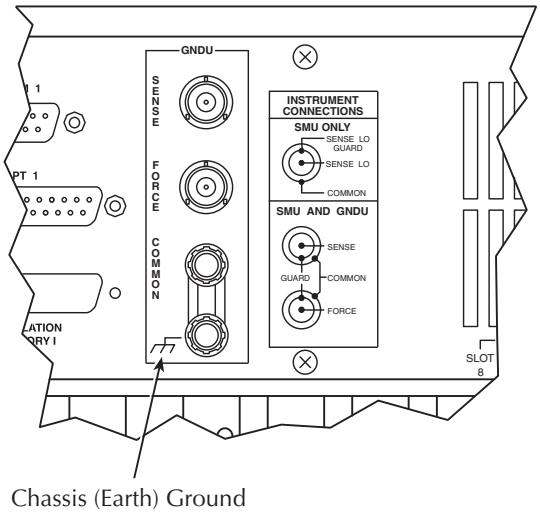
The COMMON terminal is a binding post that provides access to circuit COMMON.

NOTE *Normally, a link is connected between ground unit COMMON and chassis ground, but it may be necessary to remove the link to avoid measurement problems caused by ground loops or electrical interference. See the Interference, page 5-25 for details.*

Chassis ground

This binding post provides a convenient connecting point to system chassis ground for purposes of shielding a test fixture.

Figure 3-21
Chassis ground



Section 4

Connections and Configuration

In this section:

Introduction

This section contains detailed information about connecting and configuring the Keithley Instruments Model 4200-SCS Semiconductor Characterization System. The following topics are discussed:

- **Basic source-measure connections:** Provides basic information on making connections to the Model 4200-SMU/4210-SMU, 4200-PA, and ground unit (GNDU). Both local and remote sensing are discussed.
- **Test equipment connections:** Discusses connecting the Model 4200-SCS system to a switch matrix, test fixture, and prober.
- **Control and data connections:** Discusses connecting the Model 4200-SCS system to the test fixture or prober safety interlock, the IEEE-488 bus, printer and serial ports, and a local area network.

NOTE *Details for the Keithley pulse cards and scope card (Models 4200-SCP2 or 4200-SCP2HR) are provided in the Reference Manual, Section 11 "Pulse Source-Measure Concepts." Section 16 provides details about the Models 4220-PGU and 4225-PMU pulse cards. It also documents the Model 4225-RPM.*

Basic source-measure connections

Basic information on connecting source-measure units (SMUs), the preamp, and the ground unit to DUTs is covered in the following paragraphs. This information includes:

- Connection considerations
- SMU connections
- Preamp connections
- Using the ground unit
- Circuit COMMON connections

Connection considerations

Maximum signal limits

WARNING ***Hazardous voltages that may result in personal injury or death can be present on the signal connectors if the safety interlock is asserted. See Safety interlock connections later in this section.***

CAUTION **The maximum allowed voltage between circuit COMMON and chassis ground is ±32V DC.**

The maximum allowed voltages between the various PreAmp signals are:

- **COMMON to chassis ground: 32Vpeak**
- **GUARD to COMMON: 250 V peak**
- **SENSE or FORCE to GUARD: 40 V peak**

Shielding and guarding

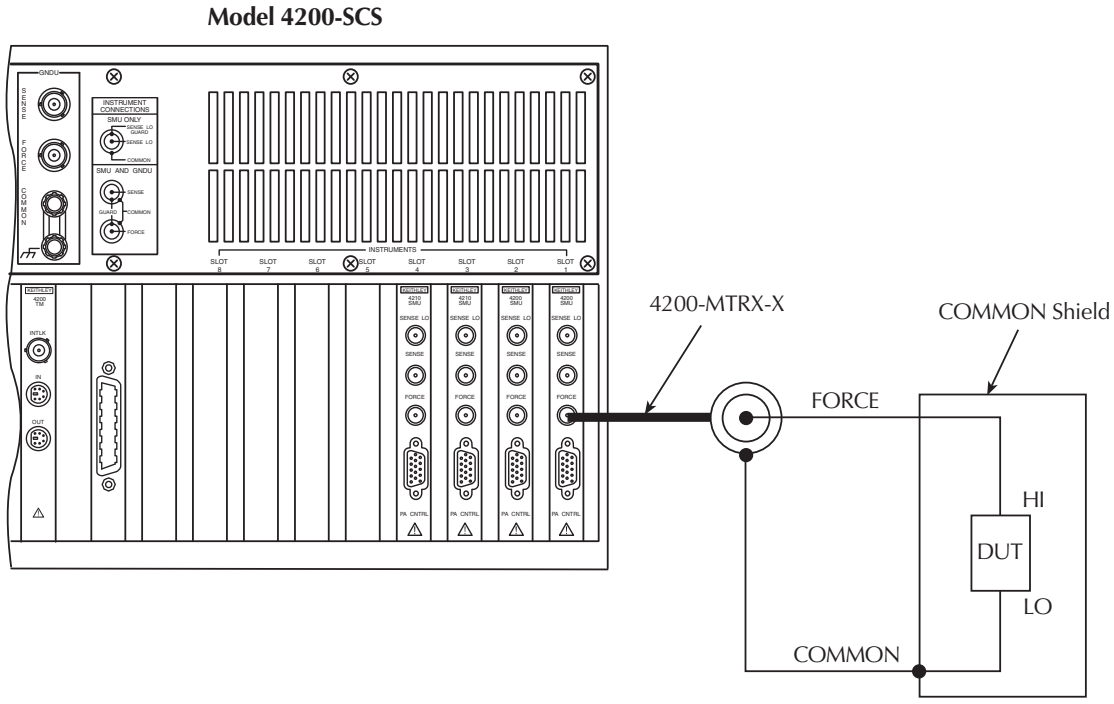
Many test situations require that the device under test (DUT) be shielded or guarded (or both) to avoid detrimental effects caused by electrostatic interference, parasitic capacitance, system leakage currents, etc.

- If the device is to be shielded (but not guarded), connect the DUT shield to COMMON (see Figure 4-1).
- If the device is to be guarded, connect the DUT shield to GUARD (inner shield of triax cable; see Figure 4-2).

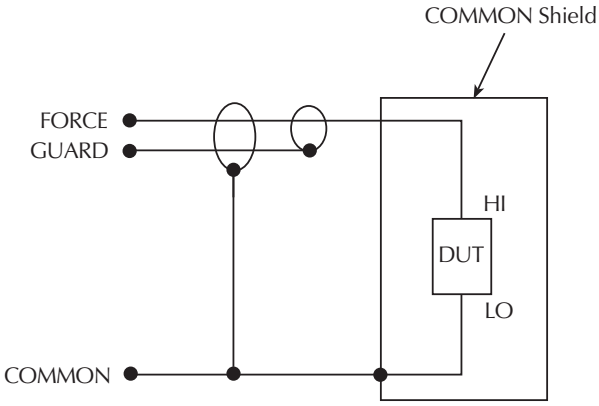
WARNING ***Hazardous voltage can be present on GUARD if the safety interlock is asserted. To avoid a shock hazard that could result in personal injury or death, surround the guard with a safety shield that is properly connected to COMMON and/or safety ground using #18AWG or larger wire.***

See Guarding in Section 5 for more information on the principles and advantages of guarding.

Figure 4-1
Device shielding

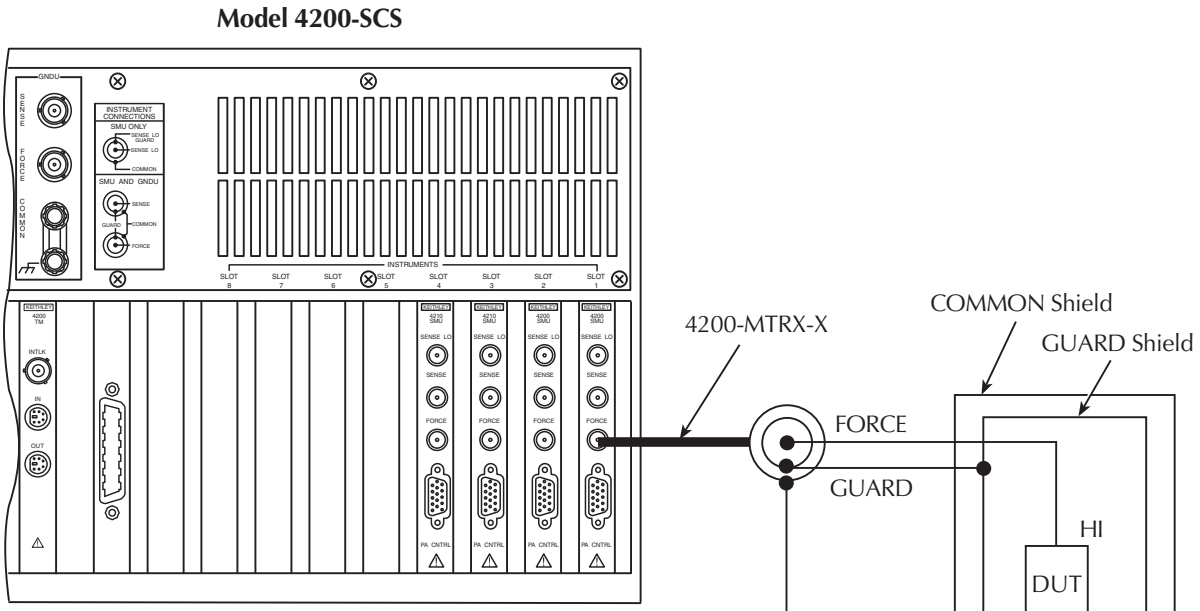


A. Connections

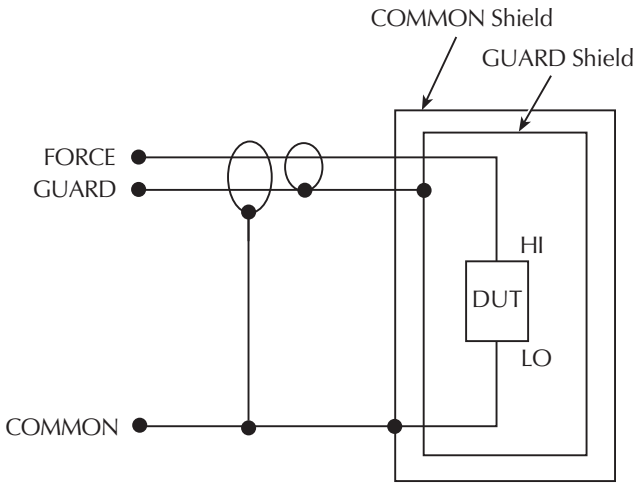


B. Equivalent Circuit

Figure 4-2
Device guarding



A. Connections



B. Equivalent Circuit

Signal integrity

To maintain signal integrity, especially at low current levels, keep the following considerations in mind when making signal connections between the Model 4200-SCS instrumentation and the DUT:

- Use only low-noise triaxial cables such as those provided with the SMU (4200-MTRX-X) and preamp (4200-TRX-X).
- Keep connecting cables as short as possible.
- Avoid flexing or vibrating connecting cables while making measurements.
- Do not touch connector insulators. Be sure to keep all connector insulators clean to minimize contamination-induced leakage currents.

Refer to Making stable measurements and Low current measurements in Section 5 for more information and these and other aspects of measurement integrity.

SMU connections

The SMU can be connected directly to the DUT with triaxial cables using either local or remote sensing, as outlined below. Remote sensing is typically used when currents exceed 1mA and the FORCE path resistance is large (around 1ohm). In this case, as much as 1mV (= 1A × 1ohm) of measurement error is generated due to FORCE path resistance. Remote sensing eliminates errors of this nature.

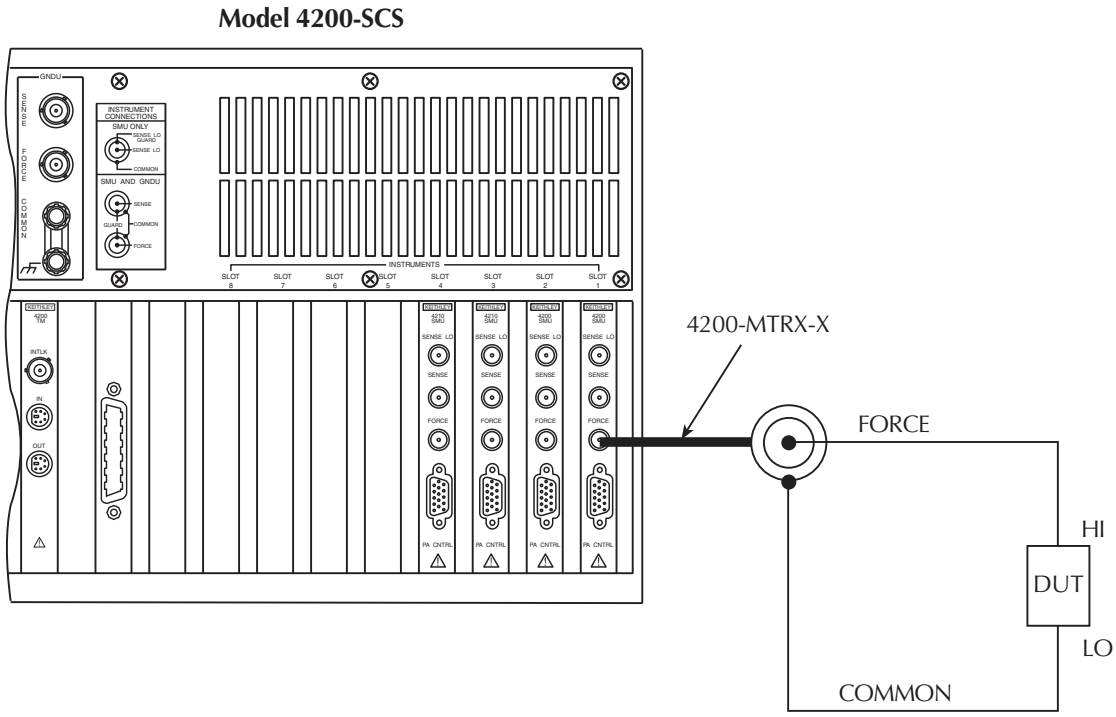
NOTE *When using more than one SMU, use the ground unit for circuit COMMON connections instead of the outer shield of the SMU terminals. Refer to Using the ground unit later in this section.*

SMU local sense connections

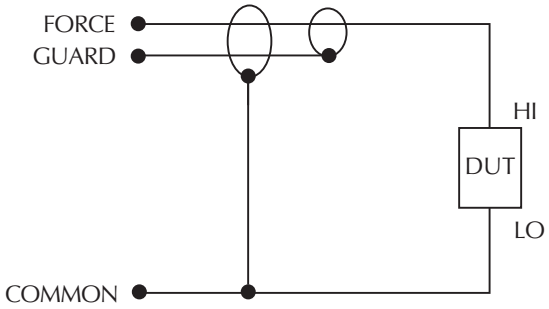
Figure 4-3 shows typical SMU connections using local sensing. Using a triax cable, make your connections as follows:

- Connect SMU FORCE (center conductor of FORCE terminal) to DUT HI.
- Connect circuit COMMON (outer shield of FORCE terminal) to DUT LO.

Figure 4-3
SMU local sense connections



A. Connections



B. Equivalent Circuit

PreAmp connections

NOTE *When using more than one PreAmp, use the ground unit for circuit COMMON connections instead of the outer shield of the PreAmp terminals (refer to Using the ground unit later in this section).*

Connecting the PreAmp to the SMU

The preamp must be connected to the SMU before use. The connecting method used depends on the mounting method (rear panel or remote). See PreAmp mounting in Section 3 for details.

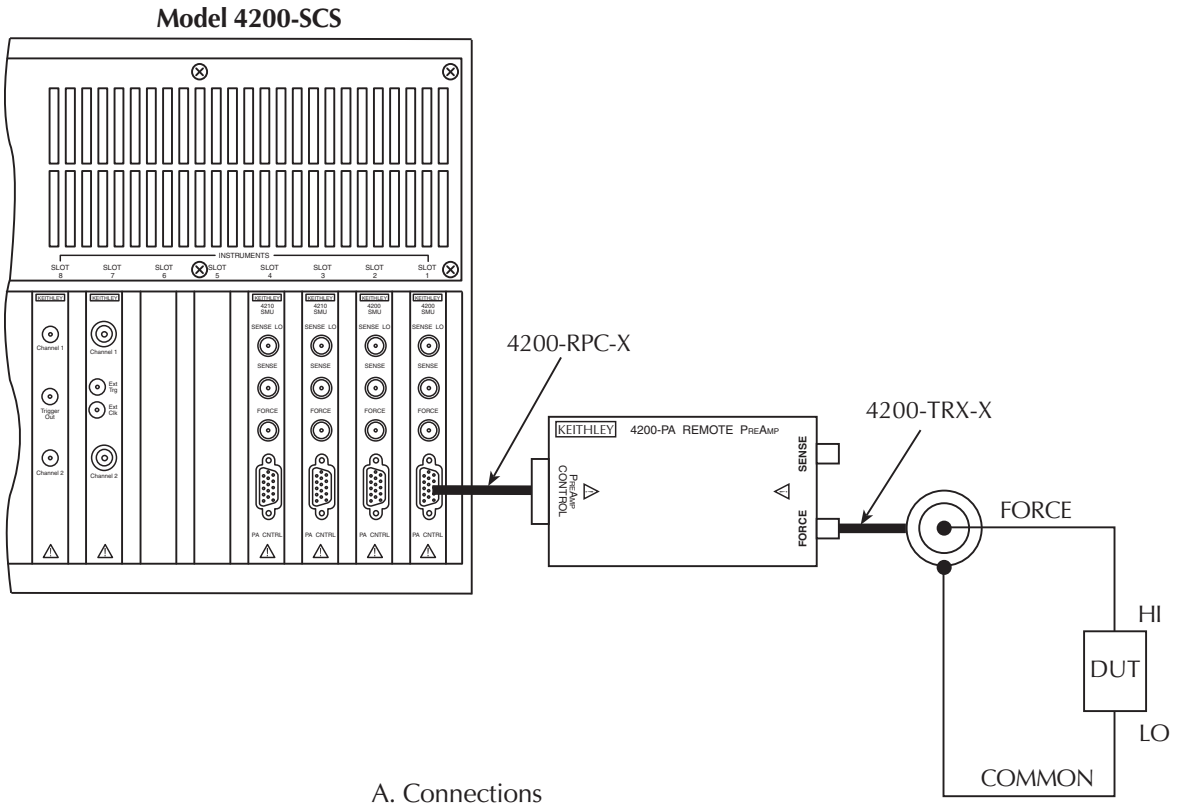
CAUTION	**Turn off system power before connecting or disconnecting the PreAmp. Failure to do so may result in SMU or PreAmp damage, possibly voiding the warranty.**

PreAmp local sense connections

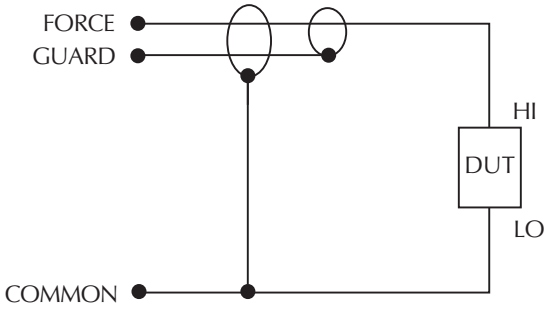
Figure 4-4 shows typical preamp connections using local sensing. Using a triax cable, make your connections as follows:

- Connect preamp FORCE (center conductor of FORCE terminal) to DUT HI.
- Connect signal COMMON (outer shield of FORCE terminal) to DUT LO.

Figure 4-4
PreAmp local sense connections



A. Connections



B. Equivalent Circuit

Using the ground unit

The ground unit (GNDU) provides convenient access to circuit COMMON via the GNDU FORCE terminal or the GNDU COMMON binding post terminal. The GNDU has a SENSE terminal as well. The SENSE LO signal of each instrument installed in the Model 4200-SCS is connected to the GNDU SENSE terminal. As a result, all SMU measurements are made relative to GNDU SENSE, which by default is connected to COMMON.

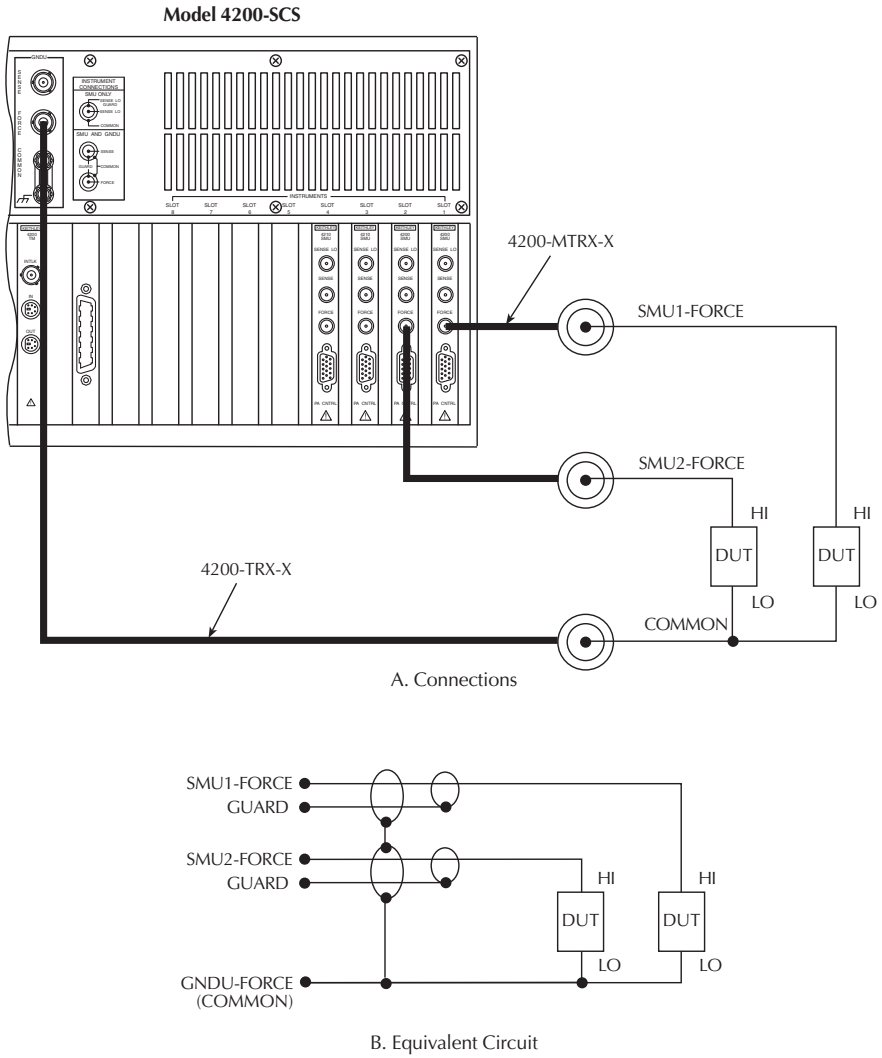
NOTE *Although the ground unit is intended for circuit COMMON connections when using multiple SMUs, it can also be used for circuit COMMON connections when using only one SMU, if desired.*

Ground unit and SMU local sense connections

Figure 4-5 shows a typical local sense connection scheme using two SMUs, two DUTs, and the ground unit. Make connections as follows:

- Connect the two SMU FORCE terminals to the two DUT HI terminals.
- Connect both DUT LO terminals together, and connect GNDU FORCE to the common DUT LO connection point.

Figure 4-5
Ground unit and SMU local sense connections



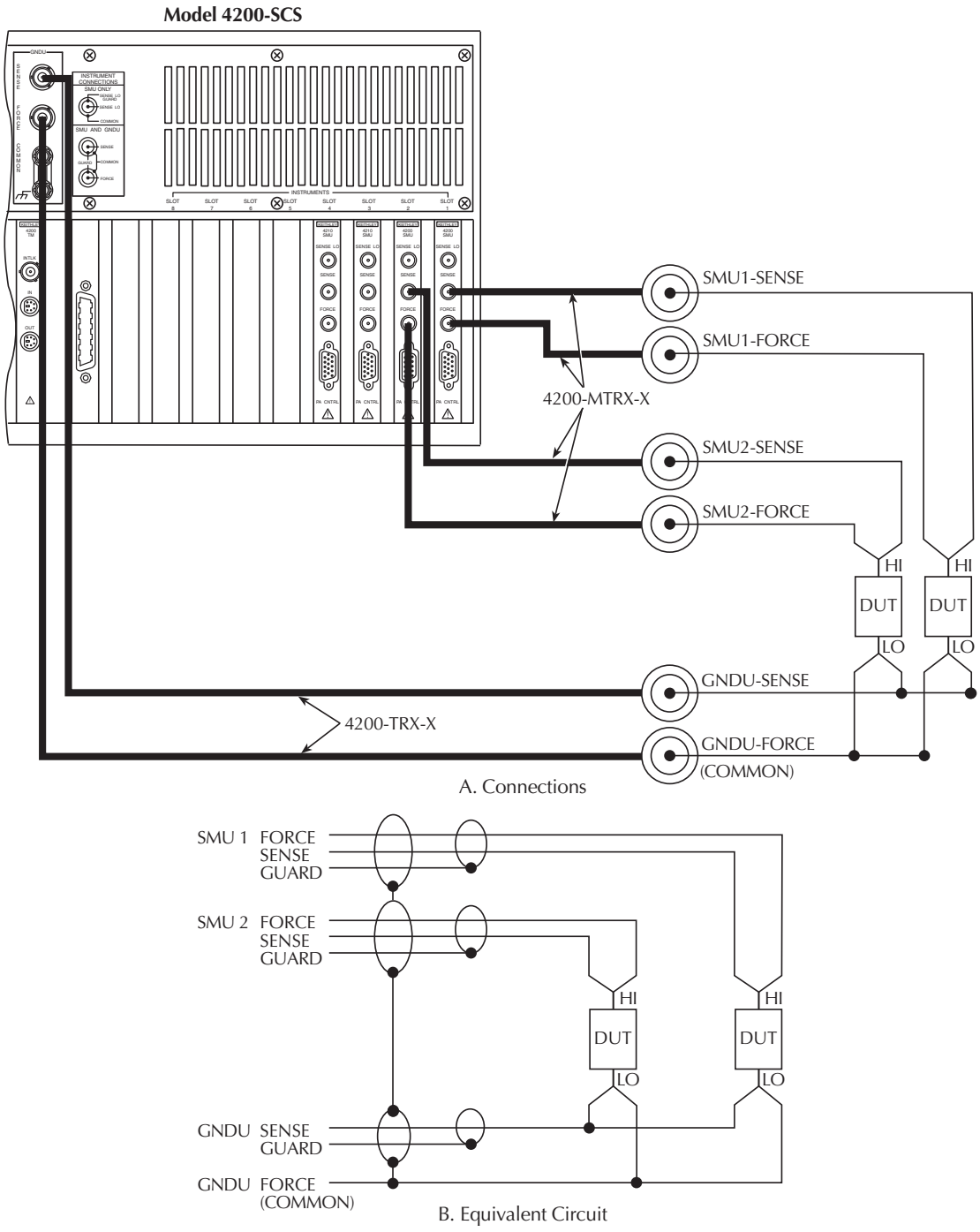
Ground unit and SMU remote sense connections

Figure 4-6 shows a typical remote sense connection scheme using two SMUs, two DUTs, and the ground unit. Make connections as follows:

- Connect the SMU FORCE and SENSE signals to the two DUT HI terminals.

- Connect both DUT LO terminals together, and connect GNDU SENSE and FORCE to the common DUT LO connection point.

Figure 4-6
Ground unit and SMU remote sense connections

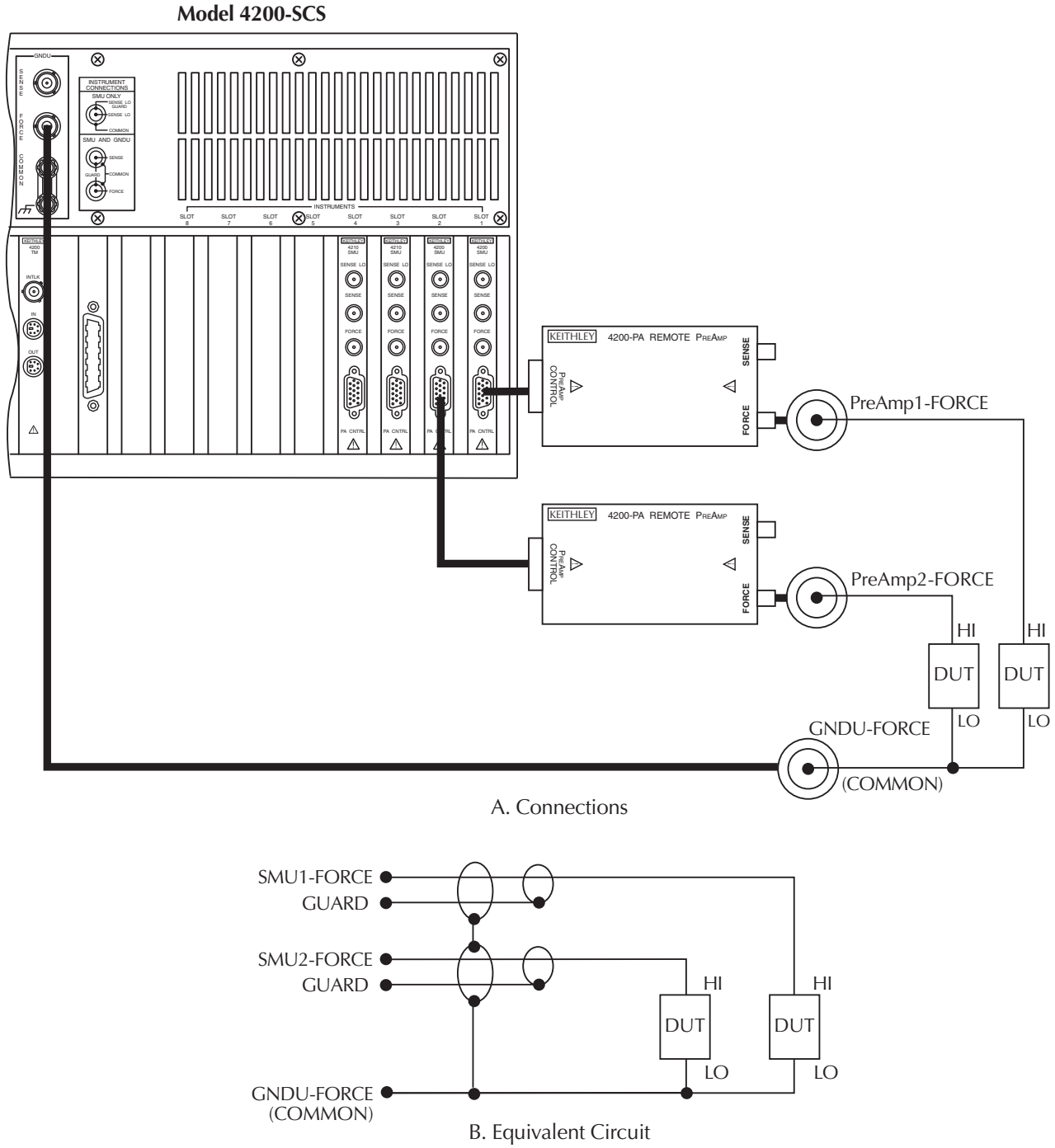


Ground unit and PreAmp local sense connections

Figure 4-7 shows a typical local sense connection scheme using two preamps, two DUTs, and the ground unit. Make connections as follows:

- Connect the two preamp FORCE signals to the two DUT HI terminals.
- Connect both DUT LO terminals together, and connect the GNDU FORCE signal to the common DUT LO connection point.

Figure 4-7
Ground unit and PreAmp local sense connections



Ground unit and PreAmp remote sense connections

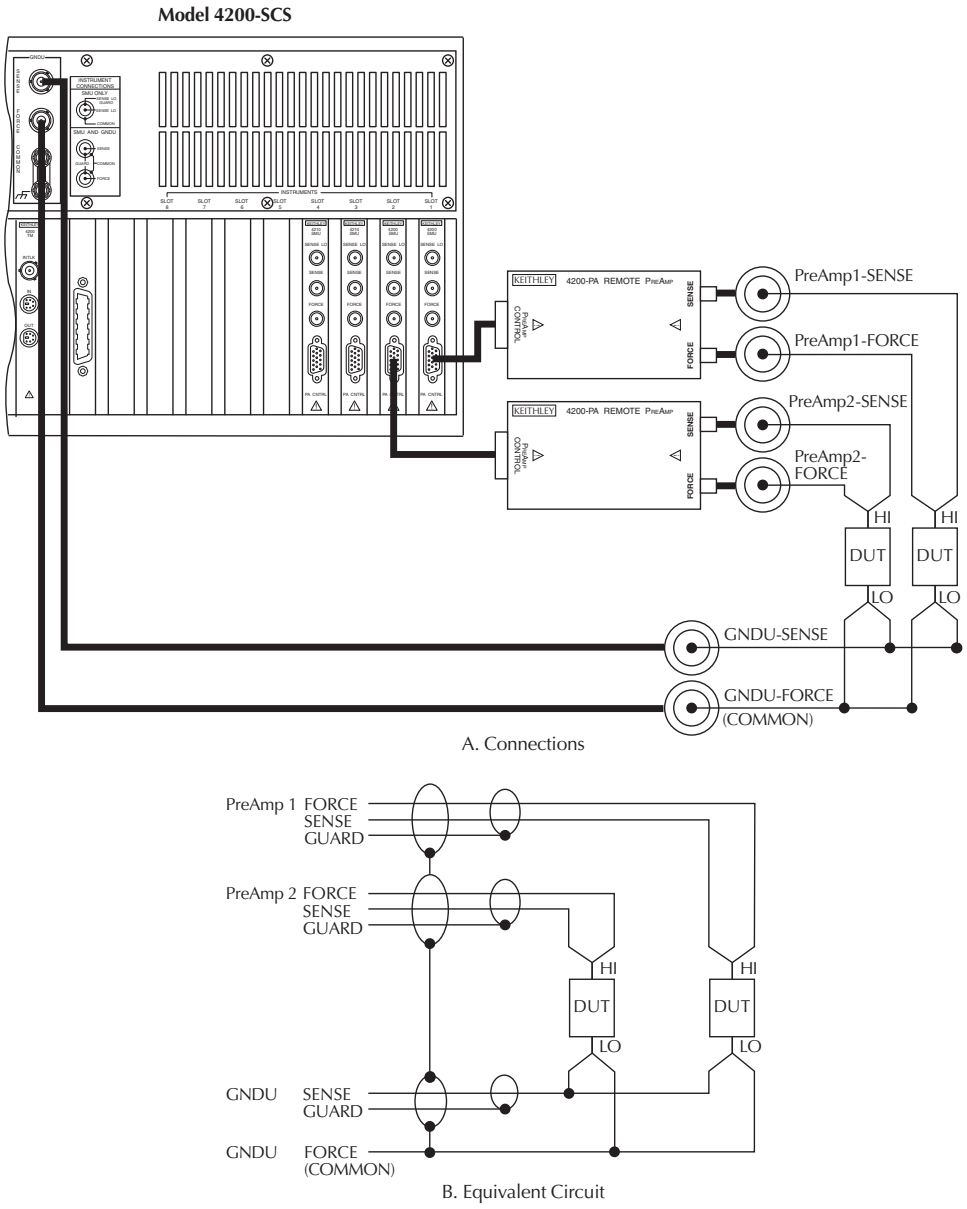
Figure 4-8 shows a typical remote sense connection scheme using two preamps, two DUTs, and the ground unit. Make connections as follows:

- Connect the preamp FORCE and SENSE signals to the two DUT HI terminals.
- Connect both DUT LO terminals together, and connect the GNDU SENSE and FORCE signals to the common DUT LO connection point.

Using the ground unit with more than two SMUs

The ground unit should also be used for circuit COMMON connections when using more than two SMUs. Make your connections using the same basic connection scheme shown in Figure 4-5, Figure 4-6, Figure 4-7, and Figure 4-8. Be sure to connect all your DUT LO terminals to the GNDU FORCE terminal (and SENSE terminal when using remote sensing).

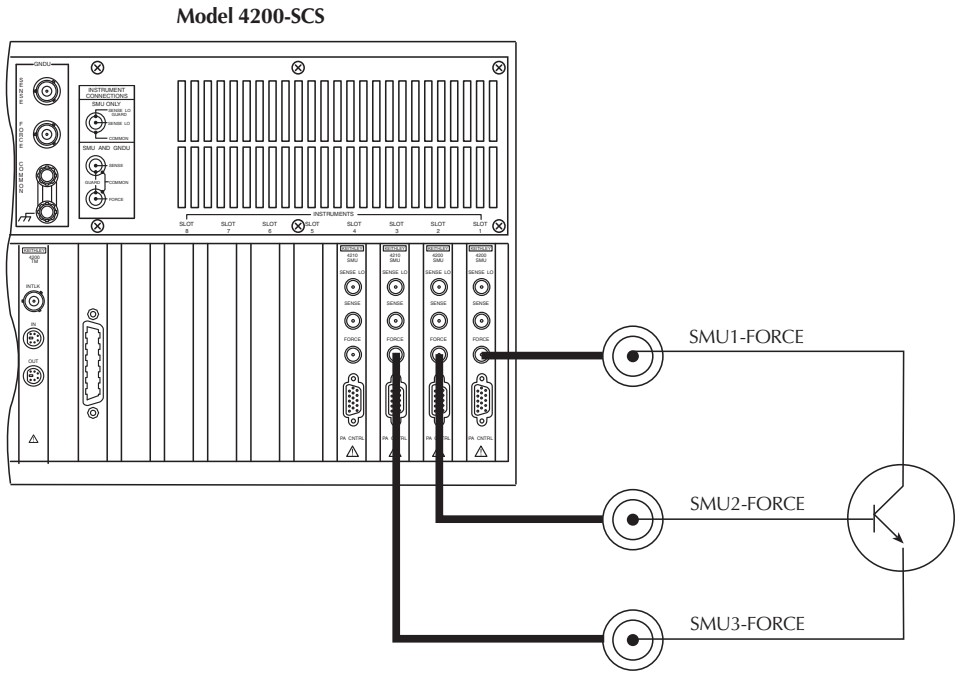
Figure 4-8
Ground unit and PreAmp remote sense connections



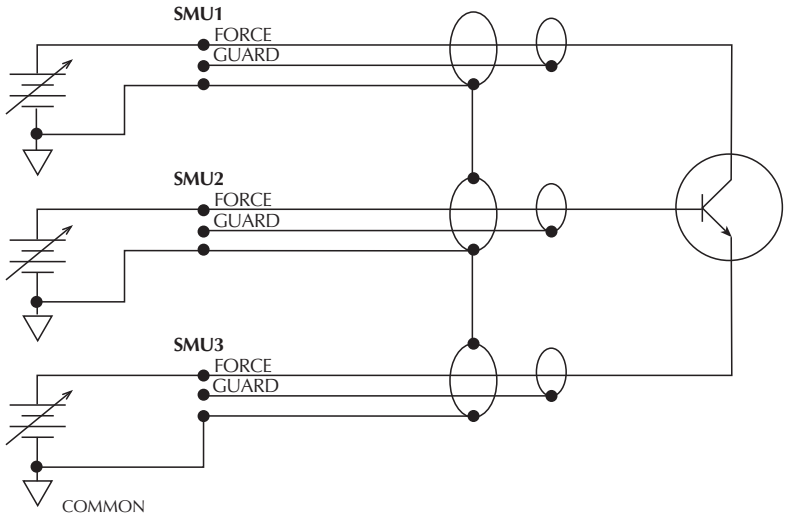
SMU circuit COMMON connections

Some test situations require SMUs to be connected to each DUT terminal. In these situations, circuit COMMON is not hardwired to any of the DUT terminals. Therefore, the SMUs must be able to internally connect circuit COMMON to their FORCE signal when the test requires a DUT terminal to be connected to COMMON. Figure 4-9 shows typical SMU connections using three SMUs to test a transistor. Any of the three SMUs could be used to provide access to circuit COMMON simply by programming it accordingly. See "Keithley Interactive Test Environment (KITE)" in Section 6 for more detailed instructions on configuring an SMU to provide a COMMON connection.

Figure 4-9
Typical SMU common connections



A. Connections



B. Equivalent circuit

Test equipment connections

The various forms of test equipment that can be used with the Model 4200-SCS include:

- Recommended connecting cables
- Switch matrix connections
- Test fixture connections
- Prober connections

Recommended connecting cables

To ensure accurate, reliable connections, use only quality, low-noise triaxial cables such as those supplied with the SMU (4200-MTRX-X) and preamp (4200-TRX-X) for all source-measure signal connections (refer to Pulsing: Source and measure options in Section 1 for a complete description of recommended triaxial cables).

NOTE *For optimum measurement accuracy, noise immunity, and settling time keep cables as short as possible.*

Switch matrix connections

A switch matrix enhances the connectivity of the Model 4200-SCS by allowing any SMU or preamp signal to be connected to any DUT pin. The following paragraphs summarize recommended switching mainframes and matrix cards, and also show typical connecting schemes with SMUs and preamps.

Switching mainframes

Table 4-1 lists recommended switching mainframes along with a brief description of each. The Keithley Instruments Models 707 and 707A hold six matrix cards, while the Keithley Instruments Models 708 and 708A hold one matrix card.

Table 4-1
Recommended switching mainframes

Mainframe	Description
Models 707 and 707A	6-slot Switching Matrix Mainframe
Models 708 and 708A	1-slot Switching Matrix Mainframe

Integrated software control for the Keithley Instruments Model 70X Switching Matrix is provided with the Model 4200-SCS KTE Interactive operating software. Refer to Using Switch Matrices in Appendix B for additional information. The Model 4200-SCS can be interfaced with switch matrices from other vendors. However, user-developed software will be needed to control these matrices from KTE Interactive. See Keithley User Library Tool (KULT) in Section 8, for additional information about developing user modules and libraries.

Recommended matrix cards

Table 4-2 summarizes recommended Keithley Instruments matrix cards, along with a brief description of each. Note that a key characteristic of these cards is low offset current to minimize the negative effects of offset currents on low-current measurements.

Table 4-2
Recommended matrix cards

Matrix card	Description
Model 7071	8 × 12 matrix, <100 pA offset current
Model 7072	8 × 12 matrix, <1pA offset current
Model 7172	8 × 12 matrix, <500fA offset current
Model 7174A	8 × 12 matrix, <100fA offset current
Model 9174	8 × 12 matrix, <100fA offset current

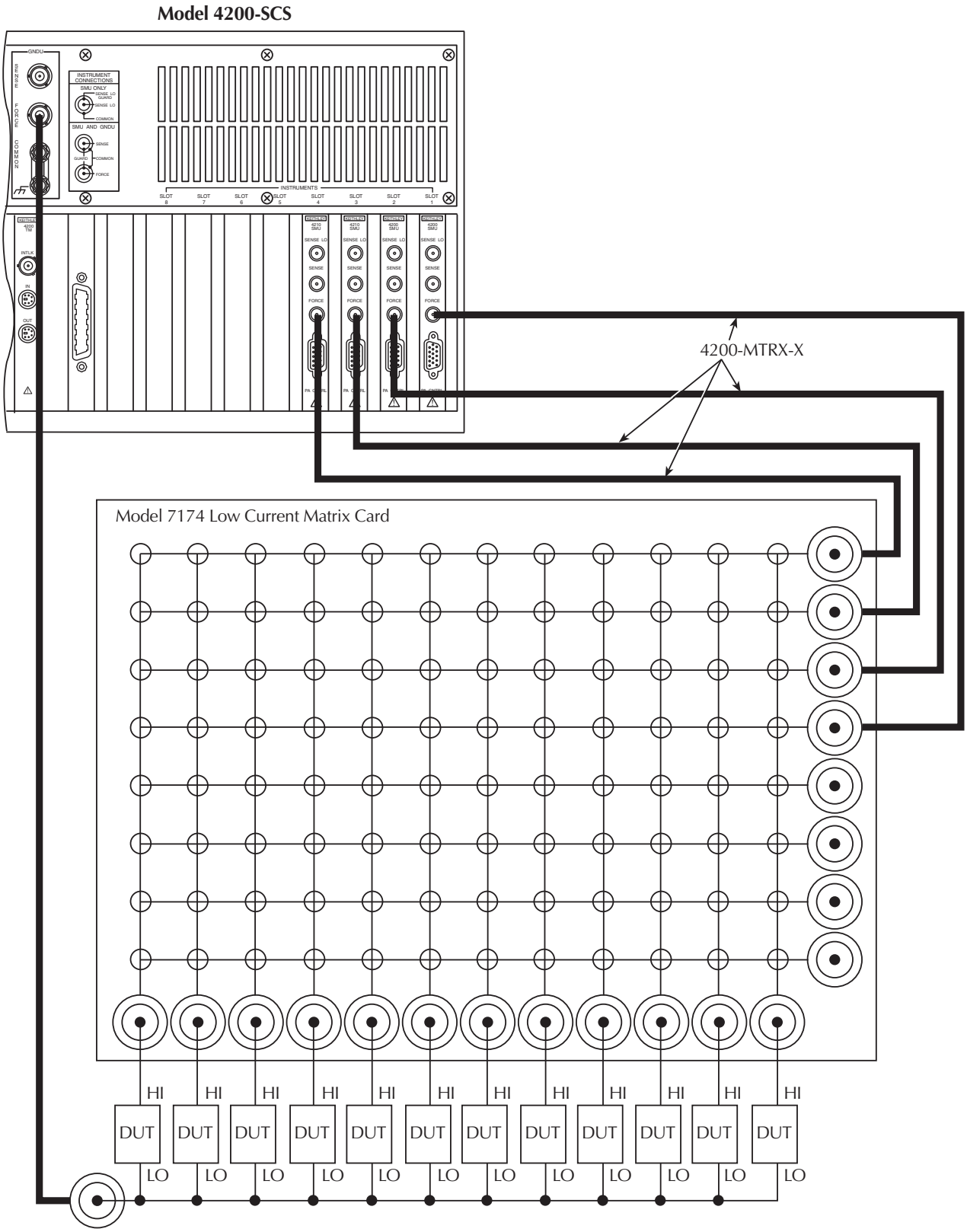
Switch mainframe control

The switch matrix is controlled by the Model 4200-SCS via the GPIB (IEEE-488) interface. Refer to IEEE-488 connections later in this section for detailed bus connection information.

Typical SMU matrix card connections

Figure 4-10 shows typical SMU matrix card connections using local sensing. Note that the four SMU FORCE terminals are connected to the matrix card rows, while the DUT HI terminals are connected to the matrix card columns. All 12 DUT LO terminals are connected together, and the DUT LO signal is connected to the ground unit FORCE terminal. Note that any SMU FORCE terminal can be connected to any DUT HI terminal simply by closing the appropriate matrix crosspoint.

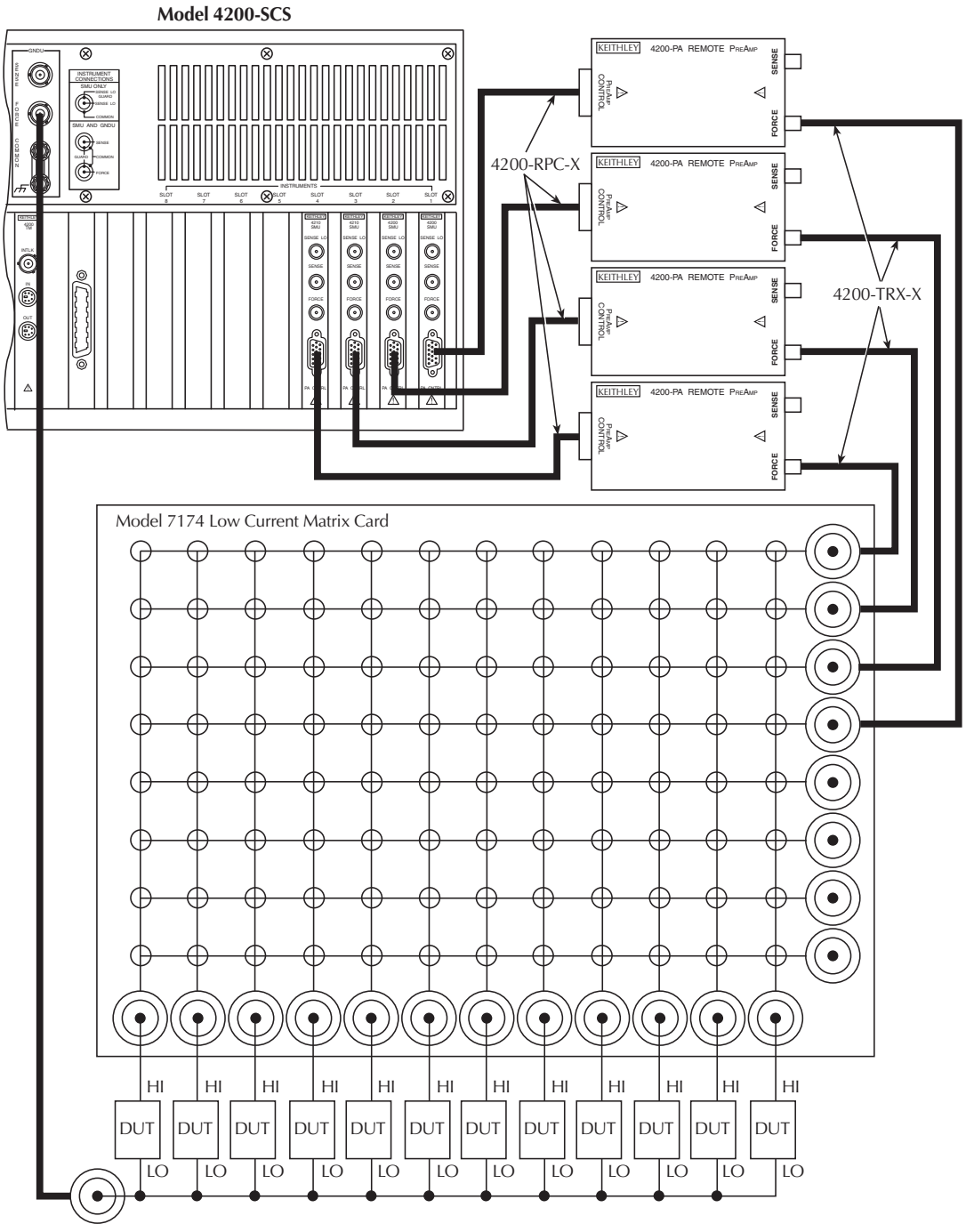
Figure 4-10
Typical SMU matrix card connections



Typical PreAmp matrix card connections

Figure 4-11 shows typical preamp matrix card connections using local sensing. This configuration is similar to the SMU configuration shown in Figure 4-10, except that preamps are added for low-current source-measure capabilities. The preamp FORCE terminals are connected to the matrix card rows, while the DUT HI terminals are connected to the matrix card columns. All 12 DUT LO terminals are connected together, and the common DUT LO signal is connected to the ground unit FORCE terminal. Again, any preamp FORCE terminal can be connected to any DUT HI terminal by closing the appropriate matrix crosspoint.

Figure 4-11
PreAmp matrix card connections



Test fixture connections

Test fixtures

Table 4-3 summarizes recommended Keithley Instruments test fixtures along with a brief description of each.

Table 4-3
Test fixtures

Test fixture	Description	DUT/fixture connections
Model 8006 Model 8007	Component test fixture Semiconductor test fixture	TO and DIP/triax connectors DIP/triax cables

WARNING ***To avoid a shock hazard that could result in personal injury or death, it is strongly recommended that you connect the safety interlock switch on the test fixture to the Model 4200-SCS interlock connector. Refer to "Safety interlock connections" later in this section for complete details.***

Prober connections

Probers

The Model 4200-SCS measurement signals can be connected to practically any commercially available wafer prober. Probers that provide triaxial connections to their probes and chuck are the easiest to interface with due to the triaxial nature of the connections on the 4200-SMU, 4210-SMU, 4200-PA, and GNDU. However, various adapters and cable kits are available from Keithley Instruments that allow the Model 4200-SCS to be adapted to any connection environment. Refer to Options and accessories in Section 1 for more information.

WARNING ***To avoid a shock hazard that could result in personal injury or death, it is strongly recommended that you connect the safety interlock switch on the probe station to the Model 4200-SCS safety interlock (INTLK) circuit. Refer to "Safety interlock connections" later in this section for complete details.***

Prober control

Semi-automatic and fully-automatic probe stations are typically controlled programmatically via an IEEE-488 or RS-232 communication interface. In this situation, the Model 4200-SCS acts as the system controller and is connected to the probe station using the appropriate communication interface. Refer to IEEE-488 connections and RS-232 connections later in this section for more information regarding interface connections.

The Model 4200-SCS facilitates automated wafer-level testing through various prober control mechanisms. Standard prober drivers are included with the Model 4200-SCS, and a number of commercially available automated probe stations are supported. The Model 4200-SCS can control supported probers without requiring the user to develop any additional software. For probers that are not supported by the standard drivers, the open architecture of the Model 4200-SCS software makes it easy to integrate prober control into the test flow by creating a user library. Refer to Keithley User Library Tool (KULT) in Section 8 for more information regarding user libraries. See the following appendices for details on enabling and configuring prober control for supported probers:

Control and data connections

The various control and data connections that interface the Model 4200-SCS to external equipment and peripherals are covered below. Topics covered include:

- Safety interlock connections
- IEEE-488 connections
- RS-232 connections
- Printer port connections
- LAN connections
- USB connections

Safety interlock connections

WARNING ***It is strongly recommended that you use the safety interlock circuit to avoid personal injury or death caused by hazardous voltages.***

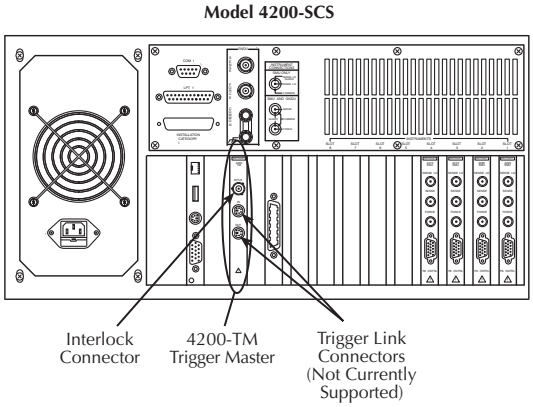
The safety interlock feature on the Model 4200-SCS should be used to avoid possible shock hazards. It provides a means by which the outputs of the Model 42XX-SMUs can be automatically placed in a safe state, regardless of the state of the Model 4200-SCS operating software. When the safety interlock signal is asserted (connected to +12V), all of the voltage ranges of the SMUs will be functional. However, when the safety interlock signal is not asserted, the ±200 V supplies of the SMUs will be disabled, limiting the nominal output to ±40 V. Under these conditions, all SMU and preamp signal terminals will be non-hazardous. Component test fixtures and probe station dark boxes typically have a safety switch that can be interfaced to the Model 4200-SCS safety interlock circuit as outlined in the following paragraphs.

Interlock connector

Figure 4-12 shows the location of the safety interlock connector.

NOTE *The Model 4200-TM IN and OUT terminals shown in Figure 4-12 are not currently supported. Connecting cables to these terminals can cause unexpected system operation.*

Figure 4-12
Interlock connector location



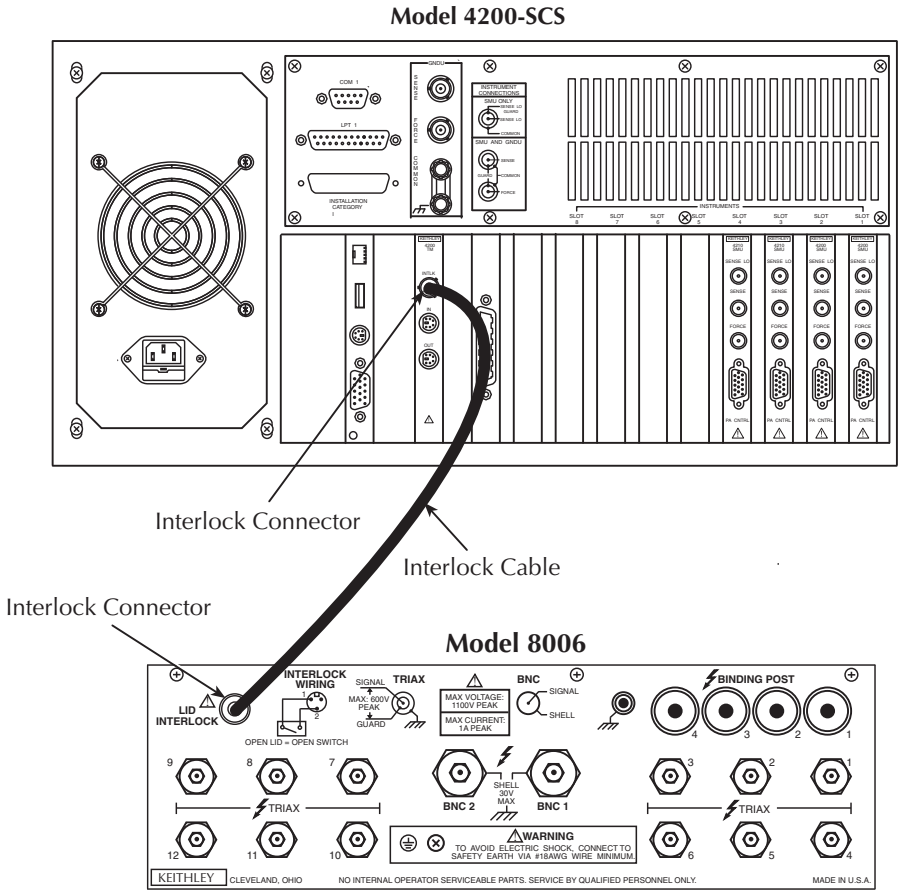
Interlock cables

Use the supplied interlock cable (236-ILC-3) or the equivalent to make interlock connections.

Typical interlock connections

Figure 4-13 shows typical interlock connections. In this example, the Model 4200-SCS is connected to a component test fixture. The test fixture has a safety interlock switch connected to its lid. When the lid is closed, the interlock circuit is closed (asserted), and SMU ±200 V ranges are enabled. Conversely, the interlock circuit is open (deasserted) when the lid is open, and SMU ±200 V ranges are disabled. A safety interlock cable is supplied with the Model 4200-SCS, allowing it to directly interface to the interlock circuits of the Keithley Instruments Models 8006 and 8007 test fixtures.

Figure 4-13
Typical interlock connections



Interlock connector wiring

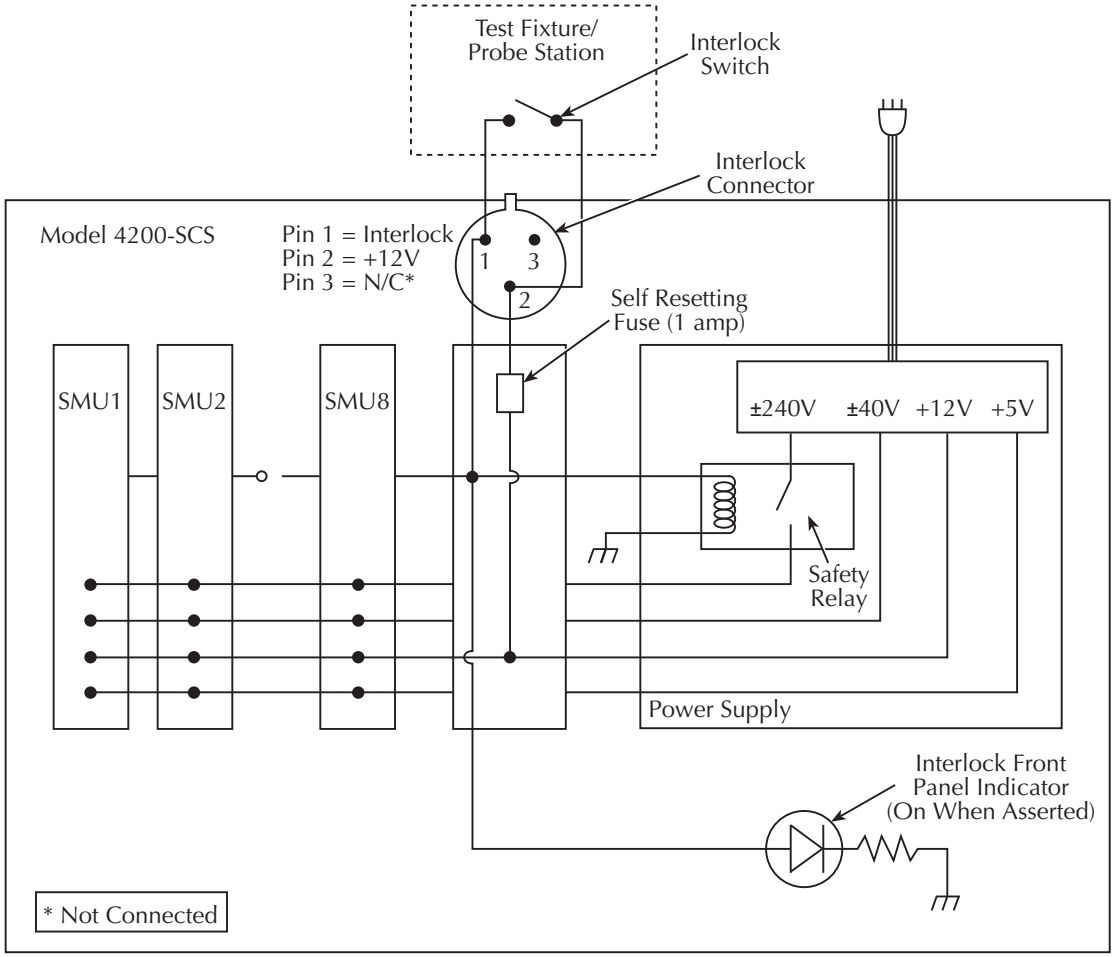
Figure 4-14 shows typical interlock connector wiring. Note that a normally open switch should be used. An open interlock condition occurs when the switch is open.

WARNING ***Ensure the interlock switch is operating correctly to assure proper, safe interlock operation.***

Configuring safety interlock operation

Set up the safety interlock as described in Control and data connections earlier in this section.

Figure 4-14
Interlock connector wiring



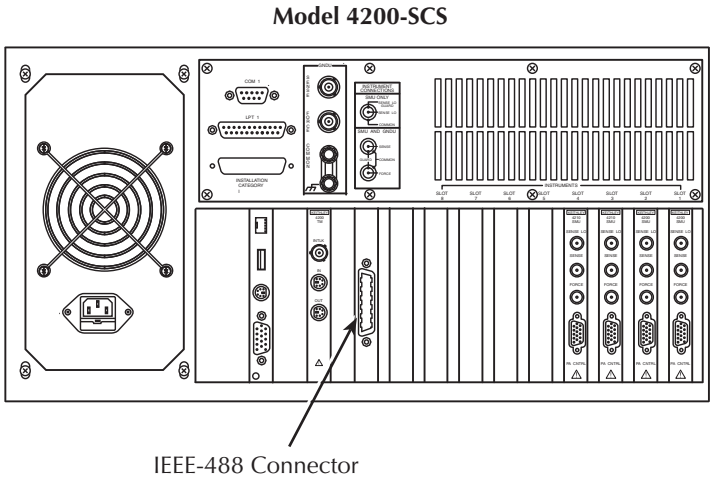
IEEE-488 connections

The built-in IEEE-488 interface allows you to interface the Model 4200-SCS to a variety of GPIB-equipped devices, such as a CV meter or switching matrix. The unit can also be configured as a GPIB slave and be controlled by an external host computer. The following paragraphs discuss the IEEE-488 connector, recommended cables, typical IEEE-488 connections, and configuring IEEE-488 controller and slave operation.

IEEE-488 connector

The Model 4200-SCS has a standard IEEE-488 connector located on the rear panel, as shown in Figure 4-15.

Figure 4-15
IEEE-488 connector location



Recommended cables

To avoid electrical interference, use only shielded IEEE-488 connecting cables such as the Keithley Instruments Models 7007-1 and 7007-2.

Configuring IEEE-488 controller operation

As previously indicated, the Model 4200-SCS can be configured to operate either as a GPIB controller or GPIB slave. The Model 4200-SCS acts as a GPIB controller when the Keithley Interactive Test Environment (KITE) is running. Refer to Keithley Interactive Test Environment (KITE) in Section 6 for more information about KITE. When operating as a controller, the Model 4200-SCS reserves primary address 0, making that address unavailable to GPIB slave devices such as GPIB switch matrices, CV meters, and automatic probe stations. Drivers for these and other instruments, typically integrated into semiconductor test systems, are included with the Model 4200-SCS. These drivers, called user libraries, permit KITE and the Model 4200-SCS to control GPIB slave devices directly. For instrumentation and equipment that is not supported by the standard user libraries, the open architecture of the Model 4200-SCS allows you to create your own user libraries using the Keithley User Library Tool (KULT). Refer to Keithley User Library Tool (KULT) in Section 8 for more information regarding the standard user libraries, KULT, and controlling external instrumentation.

Configuring IEEE-488 slave operation

The Model 4200-SCS acts as a GPIB slave when the Keithley External Control Interface (KXCI) software is running. When KXCI is running, the Model 4200-SCS can be controlled by an external computer using a command set nearly identical to the GPIB command set used to control an Agilent 4145B Semiconductor Parameter Analyzer. Refer to Keithley External Control Interface (KXCI) in Section 9 for detailed information regarding KXCI.

RS-232 connections

The built-in RS-232 port allows you to interface the Model 4200-SCS to a variety of serial devices, such as a serial printer or plotter. It can also be used to control semi-automatic probe stations and other serial equipment. The following paragraphs discuss the RS-232 connector, recommended cables, typical RS-232 connections, and configuring COM1 operation.

RS-232 connector

Figure 4-16 shows the location of the RS-232 (COM1) connector, which is a standard DB-9 (9-pin) male connector (a 9-pin-to-25-pin adapter may be used if desired). Table 4-4 summarizes both DB-9 and DB-25 connector terminals for the various RS-232 signals.

Figure 4-16
RS-232 connector location

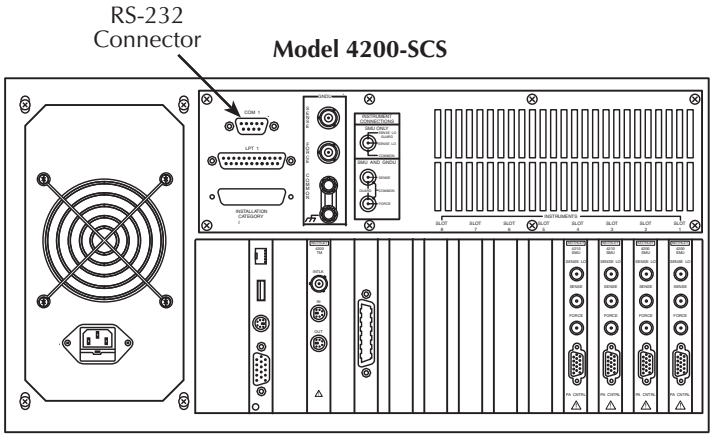


Table 4-4
RS-232 connector terminals

DB-9 pin number	DB-25 pin number	RS-232 signal
1	8	DCD, data carrier detect
2	3	RXD, receive data
3	2	TXD, transmit data
4	20	DTR, data terminal ready
5	7	GND, signal ground
6	6	DSR, data set ready
7	4	RTS, request to send
8	5	CTS, clear to send
9	22	RI, ring indicator

Recommended serial cables

To avoid electrical interference, use only properly-shielded serial cables. Shielded 25-pin cables may be used with shielded 25-pin-to-9-pin adapters where needed. Refer to Table 4-4 for equivalent pin numbers.

Configuring COM1 operation

The Model 4200-SCS can control RS-232 devices using COM1, but it cannot be connected to an external computer and controlled via COM1. The COM1 port can be used in one of three ways:

- Control a serial peripheral device such as a printer or plotter. When the Model 4200-SCS is connected to an RS-232 printer or plotter, COM1 is configured when the associated Windows® driver is installed.
- Control a semi-automatic prober. Each prober driver includes a configuration file that KITE uses when it communicates with the prober. If the prober has an RS-232 interface, this file contains all of the COM1 settings (e.g., baud rate, parity, etc.) that will be used when communicating with the prober. See Appendix G, Appendix H, Appendix I, Appendix J, Appendix K, and Appendix L for more information regarding prober control.
- Control some other type of serial instrument or equipment. For serial instrumentation or equipment that is not supported by the standard test module libraries, the open architecture of the Model 4200-SCS allows you to create your own user libraries using the Keithley User Library Tool (KULT). See Keithley User Library Tool (KULT) in Section 8 for more information regarding the standard user libraries, KULT, and controlling external instrumentation.

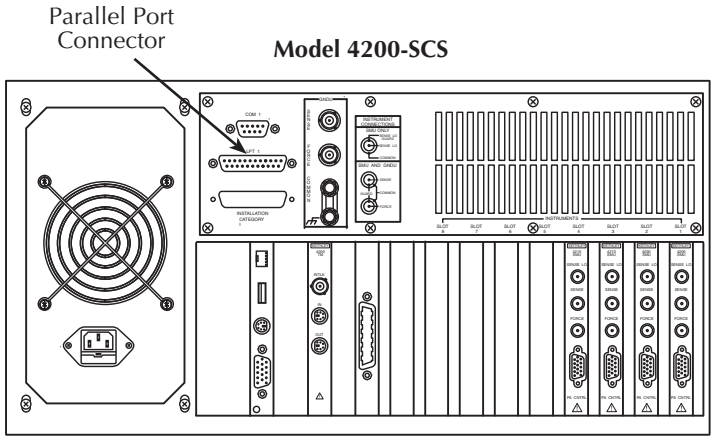
Parallel port connections

The built-in parallel port allows you to interface the Model 4200-SCS to parallel peripherals such as a printer or plotter. The following paragraphs discuss the parallel port connector, recommended cables, and typical connections.

Parallel port connector

Figure 4-17 shows the location of the parallel port connector, which is a standard female DB-25 connector.

Figure 4-17
Parallel port connector location



Recommended parallel cables

To avoid electrical interference, use a shielded parallel cable. An IEEE-1284 compliant cable is recommended for best results.

LAN connections

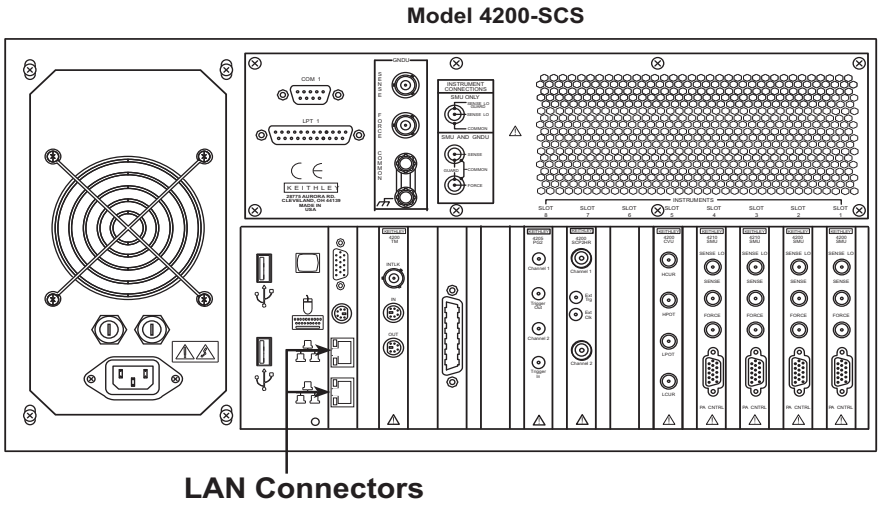
The Model 4200-SCS can be connected to an ethernet LAN so that tests and data can be easily accessed and archived. From a networking perspective, the Model 4200-SCS operates the same as any other personal computer running Windows®. The following paragraphs outline the LAN connections and recommended cables.

NOTE *The drivers for the Model 4200-SCS LAN interface are pre-installed on the system. Contact your system administrator before attempting to enable the LAN interface.*

LAN connectors

Figure 4-18 shows the location of the two LAN connectors, which are standard RJ-45 10baseT connectors intended for use with UTP (Unshielded Twisted Pair) cable.

Figure 4-18
LAN connector locations



Recommended LAN cables

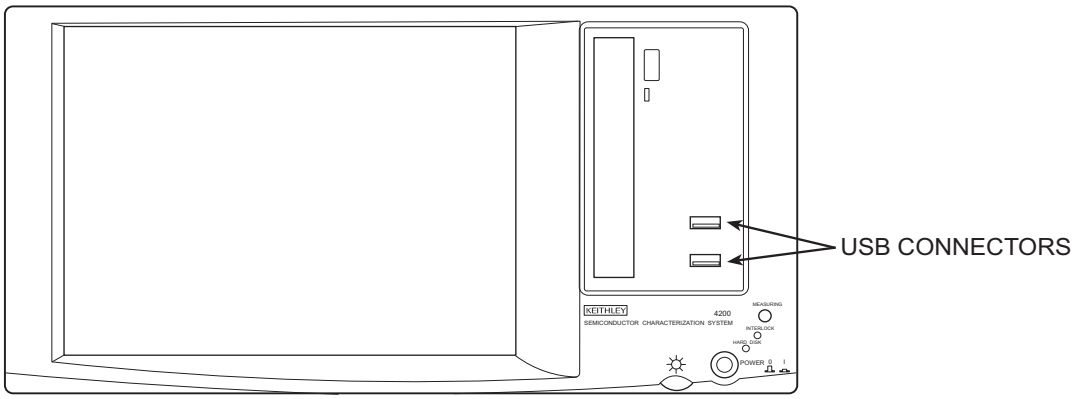
For best results, use only CAT 5 UTP cables equipped with RJ-45 connectors to connect the Model 4200-SCS to your LAN.

USB connections

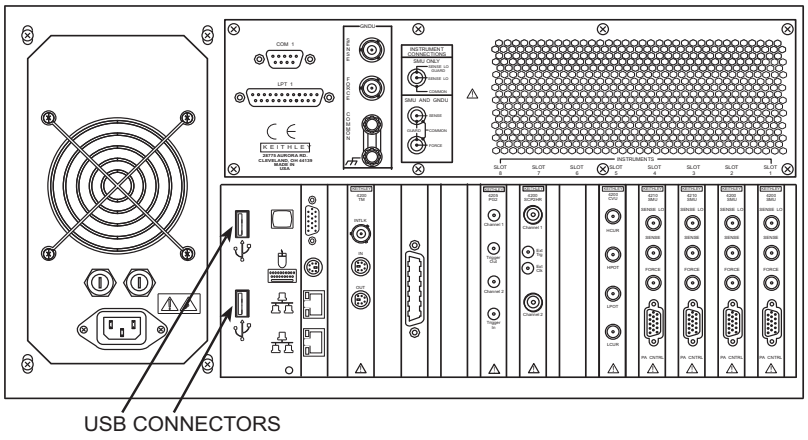
The Model 4200-SCS has 4 v2.0 USB connectors. These allow connection to USB peripherals, such as pointing devices, printers, scanners, thumb drives, external hard drives, and CD-ROMs. As shown in Figure 4-19, there are two USB connectors on the front panel and two on the rear panel.

Figure 4-19
USB connectors locations

Model 4200-SCS — Front Panel



Model 4200-SCS — Rear Panel



USB cables

Use a USB series A/B plug cable to connect a USB device to the Model 4200-SCS.

Section 5

Source-Measure Concepts

In this section:

Introduction

This section describes various source-measure concepts and is arranged as follows:

- **Guarding:** An overview of guarding, guarding concepts, guard connections, and test fixture guarding.
- **Remote sensing**: Covers an overview of sensing, sensing concepts, and sense selection.
- **Sink operation**: Provides an overview of sink operation and summarizes sink operating boundaries.
- **Source-measure configurations**: Details various source-measure circuit configurations, including Source V, Measure V, and measure-only.
- **Sweep concepts**: Covers the source-delay-measure cycle and provides an overview of sweep waveforms.
- **Making stable measurements**: Discusses various considerations when making stable measurements, including single-SMU stability, multiple-SMU stability, and avoiding oscillation.
- **Low current measurements**: Details various considerations for making low-current measurements, including leakage currents, generated currents, noise and source impedance, and voltage burden.
- **Interference**: Covers possible sources of interference such as electrostatic interference, radio frequency interference, and ground loops.

Guarding

Guarding overview

The purpose of guarding is to eliminate the effects of leakage current (and capacitance) that can exist between FORCE and COMMON, or between SENSE and COMMON. The driven GUARD is always enabled and provides a buffered voltage that is at the same level as the FORCE or SENSE HI voltage (GUARD for both SOURCE and SENSE are the same signal that is referenced in FORCE). In the absence of a driven guard, leakage in the external test circuit could be high enough to adversely affect the performance of the SMU or preamp.

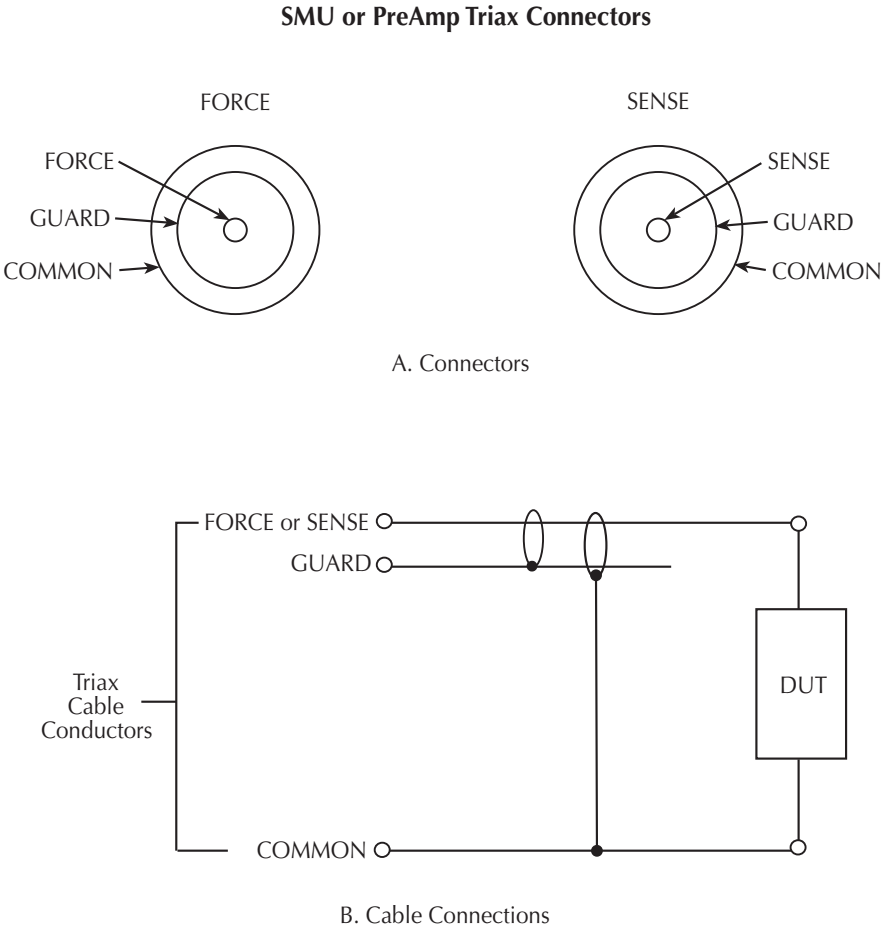
Leakage current can occur through parasitic or non-parasitic leakage paths. An example of parasitic resistance is the leakage path across the insulation in a triax cable. An example of non-parasitic resistance is the leakage path through a resistor that is connected in parallel to the DUT.

WARNING ***GUARD is at the same potential as FORCE. If hazardous voltage is present at FORCE, it is also present at the GUARD terminal. Precautions must be taken to prevent a shock hazard that could result in personal injury or death.***

Guard connections

GUARD is available at the inner shield of the FORCE and SENSE triax connectors for both the SMU and the preamp, as shown in Figure 5-1A. Figure 5-1B shows triax cable connections to the DUT. Note that GUARD is not connected in this example, but it can be routed internally to a test fixture as covered in Test fixture guarding later in this section.

Figure 5-1
GUARD connections



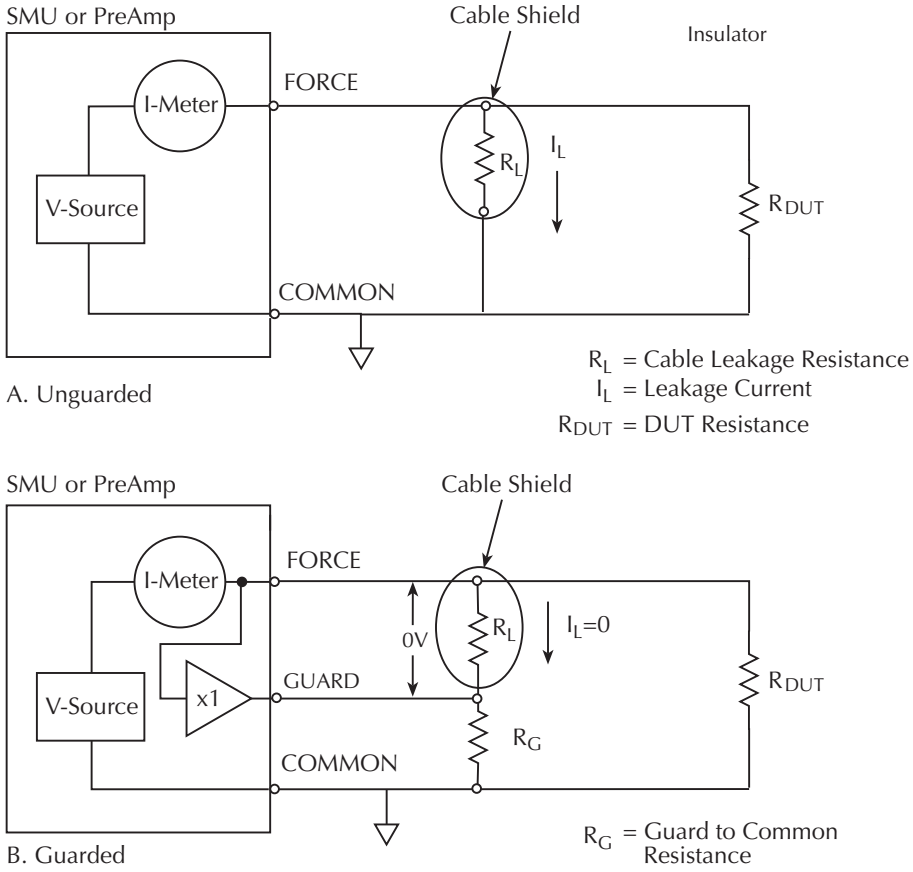
Guarding concepts

Guarding is especially important with high-impedance circuits. Consider the comparison of the unguarded and guarded circuits shown in Figure 5-2. In both cases, FORCE is connected to DUT HI, while COMMON is connected to DUT LO.

In the unguarded circuit of Figure 5-2A, the cable leakage resistance, R_L, is effectively in parallel with the DUT, creating an unwanted leakage current I_L. This leakage current may seriously affect readings, particularly at low current levels.

In the guarded circuit of Figure 5-2B, however, the cable shield is driven by a unity-gain, low-impedance amplifier (GUARD). Since the voltage across R_L is nearly 0 V, the leakage current is effectively eliminated. Current through any leakage resistance (R_G) between the shield and COMMON may be considerable, but it is of little consequence because it is supplied by the unity-gain amplifier rather than the FORCE terminal of the SMU or preamp.

Figure 5-2
Guarding concepts



Test fixture guarding

GUARD used to drive the inner shields of triax connecting cables can be routed within test fixtures. Inside the test fixture, a triax cable can be used to extend the guard near to the DUT, and the guard can be connected to a guard plate or shield that surrounds the DUT. The center conductor of the cable is used for FORCE or SENSE, the inner shield is used for GUARD, and the outer shield is COMMON.

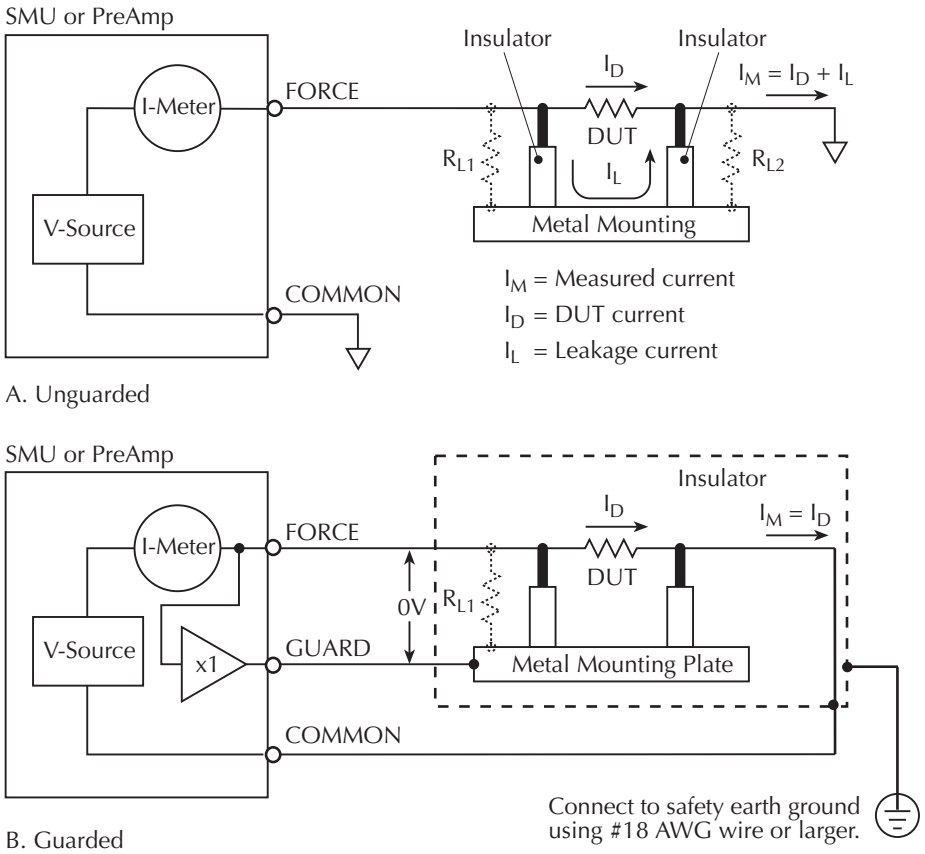
WARNING ***To prevent injury or death, a safety shield must be used to prevent physical contact with a guard plate or guard shield that is at a hazardous potential (>30 Vrms or 42.4V peak). This safety shield must completely enclose the guard plate or shield, and must be connected to common and/ or to safety earth ground. Figure 5-2B shows the metal case of a test fixture being used as a safety shield. It is strongly recommended that you use the test fixture safety interlock. Refer to Safety interlock connections in Section 4.***

Figure 5-3 shows how GUARD can eliminate leakage current through the insulators in a test fixture. In Figure 5-3A, leakage current (I_L) flows through the insulators (R_{L1} and R_{L2}) to COMMON, adversely affecting the low-current (or high-resistance) measurement of the DUT.

In Figure 5-3B, the driven GUARD is connected to the metal guard plate for the insulators. Since the voltage on either end of R_{L1} is the same (0 V drop), no current can flow through the leakage resistance path. As a result, the SMU or preamp measures only the current through the DUT.

NOTE *The GUARD signal has an output impedance of 100 k Ω and is, therefore, effective only when connected to high-impedance loads.*

Figure 5-3
Test fixture guarding



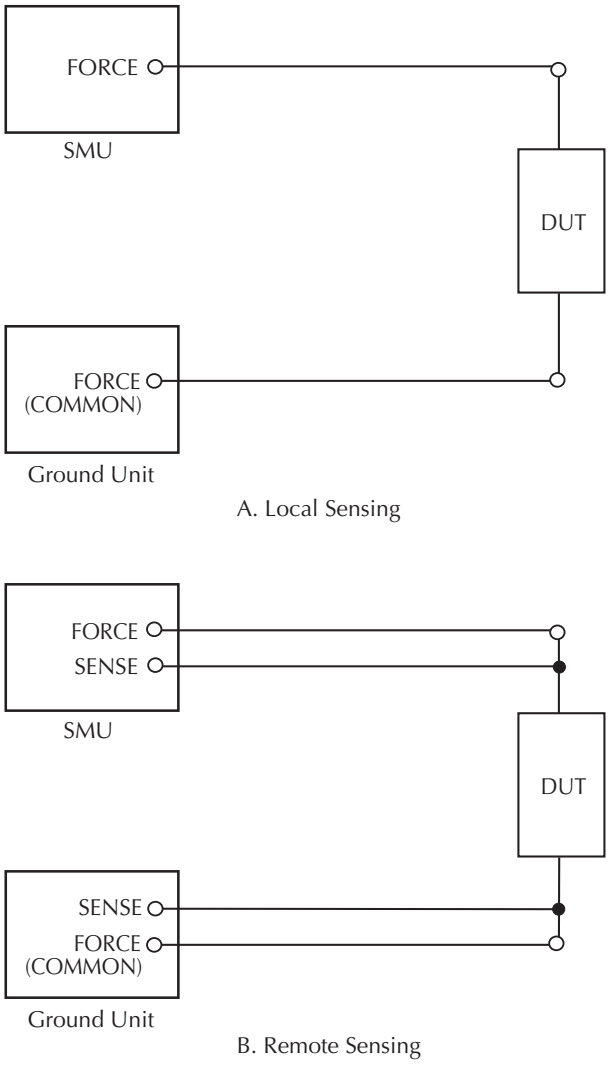
Remote sensing

Sensing overview

As shown in Figure 5-4, there are two types of sensing: local and remote. With local sensing (Figure 5-4A), only two terminals are connected to the DUT: SMU FORCE and GROUND UNIT: FORCE (COMMON). With remote sensing (Figure 5-4B), both SENSE terminals are connected to the DUT, along with both FORCE terminals.

NOTE *See Basic source-measure connections in Section 4 for detailed information on various connection methods. GUARD connections are not shown in Figure 5-4 for the sake of clarity.*

Figure 5-4
Sensing overview



Sense selection

The sensing method is automatically selected depending on the connection method used. To use local sensing, connect only SMU FORCE and ground unit FORCE (COMMON) to the DUT (Figure 5-4A). To use remote sensing, add the SENSE connections shown in Figure 5-4B.

Sensing concepts

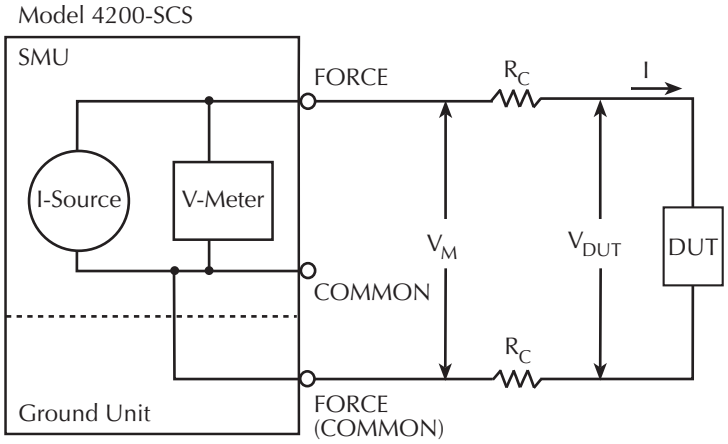
Local sensing

Measurements made on devices with impedances above approximately 1k Ω are generally made using the local sensing method shown in Figure 5-5. The SMU test current is forced through the test leads and the DUT being measured, developing a voltage across the device (V_{DUT}). The SMU then measures the voltage across the DUT (V_M) through the same set of test leads.

The main problem with the local sensing method with low-impedance DUTs is the cable resistance (R_C), as well as the connection resistance (such as matrix crosspoint resistance or prober-to-IC pad resistance), can be as high as 1 Ω. Since the test current I causes a small, but significant

voltage drop across the cable resistance, the voltage measured by the SMU (V_M) will not be exactly the same as the voltage directly across the DUT (V_{DUT}), and considerable error can result. Typical cable resistances lie in the range of 1m Ω to 100 m Ω, so it may be difficult to obtain accurate local sensing measurements with DUT resistances below 100 Ω to 1 k Ω, depending on the magnitudes of both cable resistance and contact resistance.

Figure 5-5
Local sensing



R_C = Cable Resistance

I = Test Current Through DUT

V_M = Measured Voltage

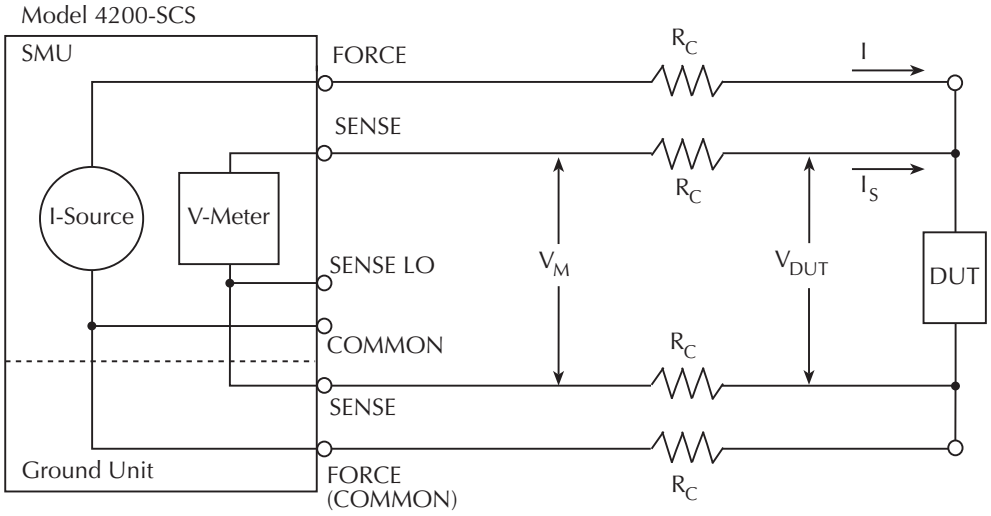
V_{DUT} = Voltage Across DUT

$V_{DUT} < V_M$ because of I Through R_C

Remote sensing

Due to the limitations of local sensing, the remote sensing method shown in Figure 5-6 is generally preferred for measurements on low-impedance DUTs. With this configuration, the test current I is forced through the DUT through one set of test cables, while the voltage across the DUT is measured through a second set of sense cables. Although some small current (I_S) may flow through these sense cables, it is usually negligible (typically pA or less) and can generally be ignored for all practical purposes. Since the voltage drop across the sense cables is negligible, the voltage actually measured by the SMU (V_M) is essentially the same as the voltage across the DUT (V_{DUT}).

Figure 5-6
Remote sensing



R_C = Cable Resistance
I = Test Current Through DUT
I_S = Sense Current (Negligible)
V_M = Measured Voltage
V_{DUT} = Voltage Across DUT
V_{DUT} = V_M because of Negligible I_S

Sensing considerations

Local sensing is adequate for many test and measurement situations. However, for maximum accuracy, it is recommended that you use remote sensing for the following source-measure conditions:

- Test circuit impedance is <1 k Ω.
- Maximum V-Source, and/or V-Measure accuracy are required

NOTE *Specified accuracies for both source and measure are achieved using remote sensing.*

Sink operation

Sink overview

When operating as a sink (V and I have opposite polarity), the SMU is dissipating power rather than sourcing it. An external source (such as another SMU) or an energy storage device (like a capacitor) can force operation into the sink region.

For example, if a second SMU that is sourcing +12V is connected to the first SMU programmed for +10 V, sink operation for the first SMU will occur in the second quadrant (source +V and measure -I).

CAUTION **When using the I-Source as a sink, always set the voltage compliance to a level that is higher than the external voltage level. Failure to do so could damage the SMU or PreAmp due to excessive current that will flow into the unit.**

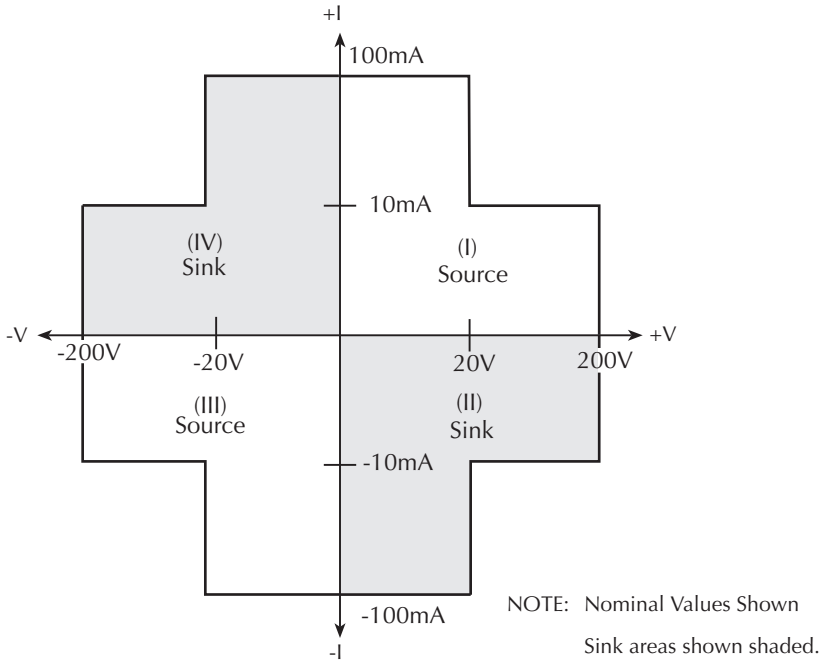
Sink operating boundaries

Sink operating boundaries for the Models 4200-SMU and 4210-SMU are shown in Figure 5-7 and Figure 5-8, respectively. Note that sink boundaries are shown shaded, while source boundaries are unshaded.

Model 4200-SMU sink boundaries

Nominal Model 4200-SMU boundaries are shown in Figure 5-7. Note that actual boundaries are 210 V at 10.5 mA or 21V at 105 mA.

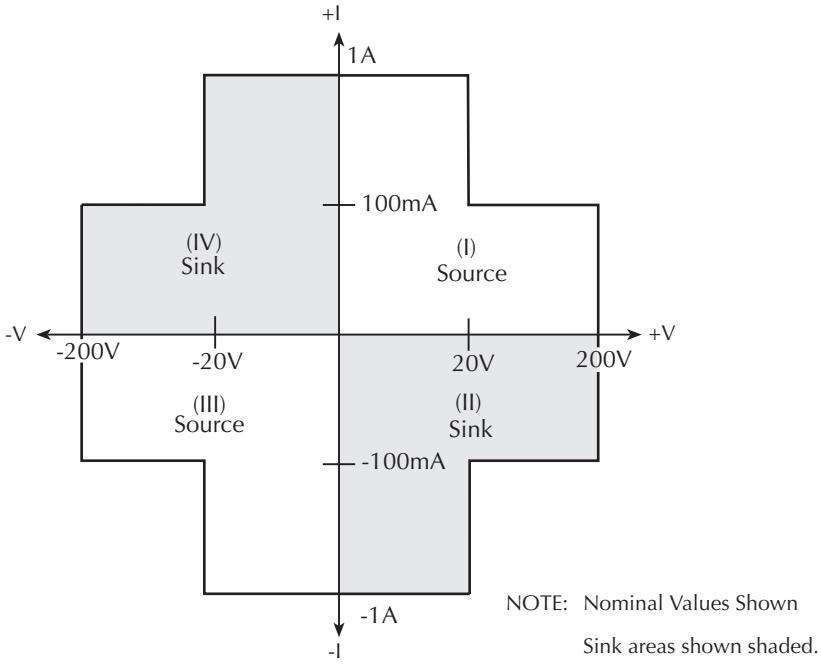
Figure 5-7
Model 4200-SMU sink operating boundaries



Model 4210-SMU sink boundaries

Nominal Model 4210-SMU sink boundaries are shown in Figure 5-8. Actual boundaries are 210 V at 105 mA or 21V at 1.05 A.

Figure 5-8
Model 4210-SMU sink operating boundaries



Source-measure configurations

Source I, measure V or I

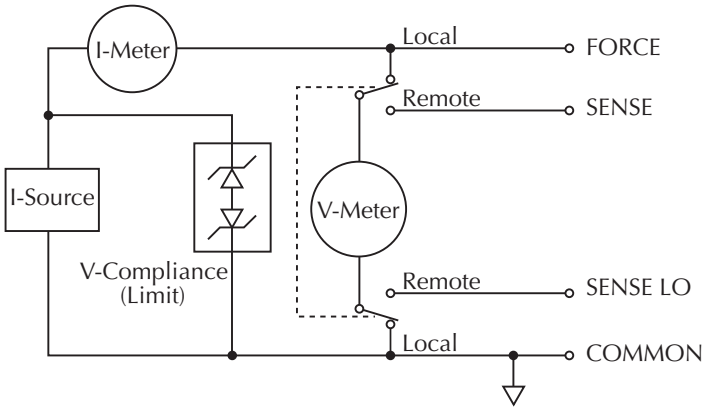
When configured to source current (I-Source) as shown in Figure 5-9, the SMU functions as a high-impedance current source with voltage limit capability that can measure current (I-Meter) or voltage (V-Meter). The compliance circuit limits the output voltage to the programmed value.

For voltage measurements, the SENSE selection (local or remote) determines where the measurement is made. In local SENSE, voltage is measured at the FORCE and COMMON terminals of the SMU.

In remote SENSE, voltage can be measured directly at the DUT using the SENSE and SENSE LO terminals. This method eliminates any voltage drops that may be in the test cables or connections between the SMU or preamp and the DUT.

NOTE *The current source does not require or use the SENSE leads to enhance current source accuracy.*

Figure 5-9
Source I, measure V configuration



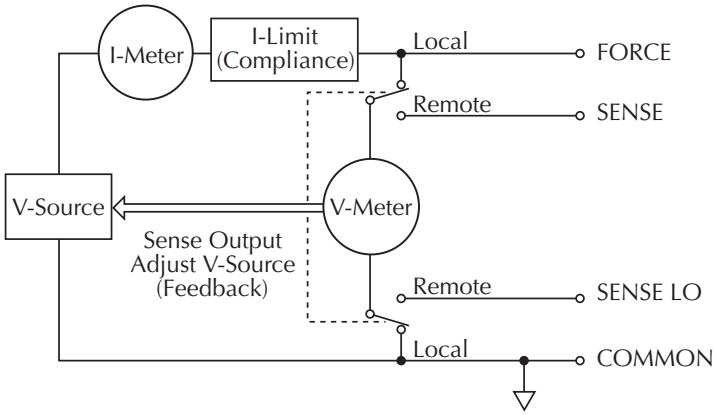
Source V, measure I or V

When configured to source voltage (V-Source) as shown in Figure 5-10, the SMU functions as a low-impedance voltage source with current limit capability and can measure current (I-Meter) or voltage (V-Meter). The compliance circuit limits the current to the programmed value.

Sense circuitry is used to continuously monitor the output voltage and make adjustments to the V-Source as needed. The V-Meter senses the voltage at the FORCE and COMMON terminals (local SENSE) or at the DUT (remote SENSE using the SENSE and SENSE LO terminals) and compares it to the programmed voltage level. If the sensed level and the programmed value are not the same, the V-Source is adjusted accordingly. Remote SENSE eliminates the effect of voltage drops in the test cables, ensuring that the exact programmed voltage appears at the DUT.

NOTE *Feedback to the V-Source is an analog function. The V-Source is adjusted to compensate for IR drop between the V-Source and the sense point.*

Figure 5-10
Source V, measure I configuration



Measure only (V or I)

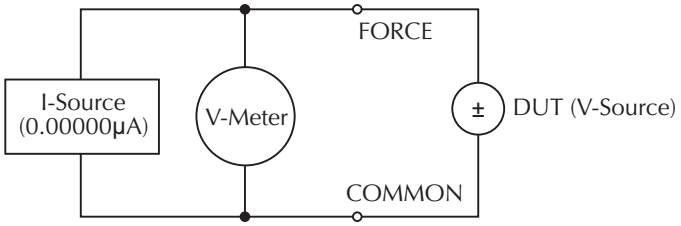
Figure 5-11 shows the configurations for using the SMU exclusively as a voltmeter or ammeter. As shown in Figure 5-11A, the SMU is configured to measure voltage only by setting it to source 0A and measure voltage. See caution below.

NOTE *The configuration shown in Figure 5-11A applies to SMU voltmeter unit (VMU) operation. See Hardware features and capabilities in Section 1 for VMU operating characteristics.*

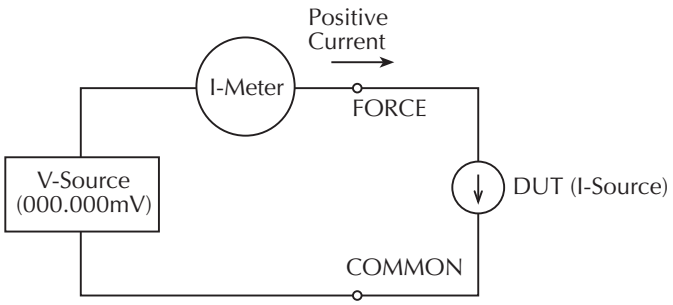
In Figure 5-11B, the SMU is configured to measure current only by setting it to source 0 V and measure current. Note that in order to obtain positive (+) readings, conventional current must flow from FORCE to COMMON.

CAUTION **For measure V, the voltage compliance should be set higher than the measured voltage. For measure I, the current compliance should be set higher than the measured current.**

Figure 5-11
Measure only configurations



A. Measure voltage only



NOTE: Positive current flowing out of FORCE results in positive (+) measurements.

B. Measure current only

NOTE: Use local sensing

Sweep concepts

Source-delay-measure cycle

Although the SMU can be used for static source or measure operation, SMU operation usually consists of a series of source-delay-measure (SDM) cycles (see Figure 5-12) as part of a sweep (refer to Sweep waveforms). During each SDM cycle, the following occurs:

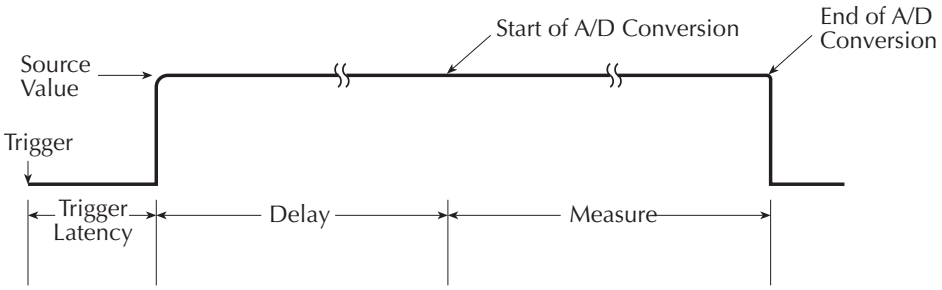
1. The source output level is set.
2. There is a wait for the source delay.
3. The measurement is made.

The delay phase of the SDM cycle, which is programmed by software, allows the source and external circuitry to settle before the measurement is performed. Although the source itself settles quite quickly (provided the unit is not in compliance), external V or I settling may take considerably longer due to interaction between the DUT and the SMU.

When there is more capacitance seen at the output, there will be more settling time required for the source signal. The actual delay period needed can be calculated or determined by trial and error. For purely resistive loads and at higher current levels, the programmable delay can be set to a minimum.

The measure time depends on the selected integration period, and it also can be extended by autorange.

Figure 5-12
SDM cycle



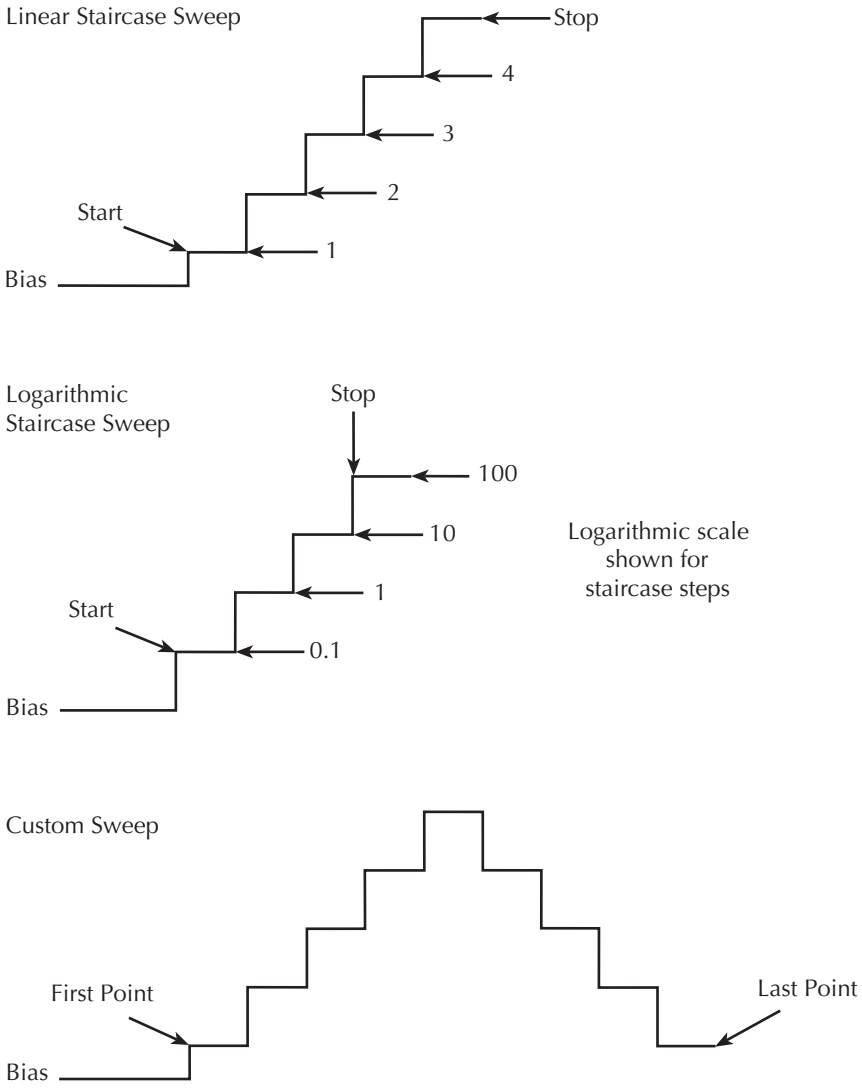
Sweep waveforms

There are three general sweep types: linear staircase, logarithmic staircase, and custom (refer to Figure 5-13). The linear staircase sweep goes from the start level to the stop level in equal linear steps. The logarithmic staircase sweep is similar, except it is done on a log scale with a specified number of steps per decade. The custom sweep lets you construct your own sweep by specifying the number of measure points and the source level at each point.

An SDM cycle is performed on each step (or point) of the sweep; one measurement will be performed at each step (level). The time spent at each step depends on how the SDM cycle is configured for such aspects as the programmed delay.

Typical applications for staircase sweeps include: I-V curves for two- and three-terminal semiconductor devices, characterization of leakage versus voltage, and semiconductor breakdown.

Figure 5-13
Sweep waveforms



Making stable measurements

NOTE *The Models 4200-SMU and 4210-SMU have been designed to be stable under a wide variety of measurement situations; however, the following information has been provided for those who may encounter instability problems.*

Single SMU stability considerations

Current source stability

Driving inductive loads can cause current source instability; current source instability almost never occurs in semiconductor applications.

Voltage source stability

A SMU that is sourcing voltage is stable when driving capacitive loads up to 10nF. However, at the lower current measurement ranges, large capacitive loads may increase settling time and may cause overshoot and ringing. To reduce this effect, do one or both of the following:

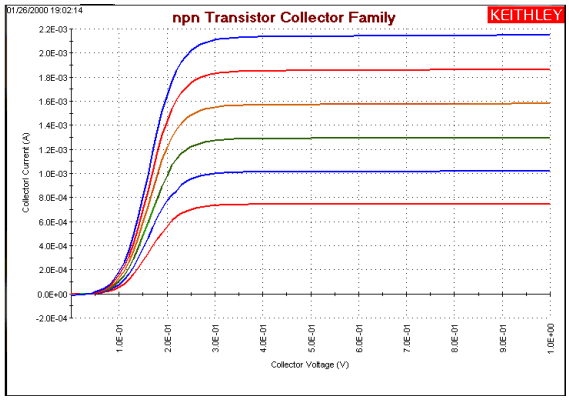
- Increase the measurement **Delay Factor** for the test being performed, using the KITE ITM timing panel (refer to Configuring the Speed and Timing settings in the ITM Definition tab in Section 6).
- Add a small resistor in series with the capacitive load. Choose a resistor that provides an RC time constant of 1ms to 10 ms.

Multiple SMU stability considerations

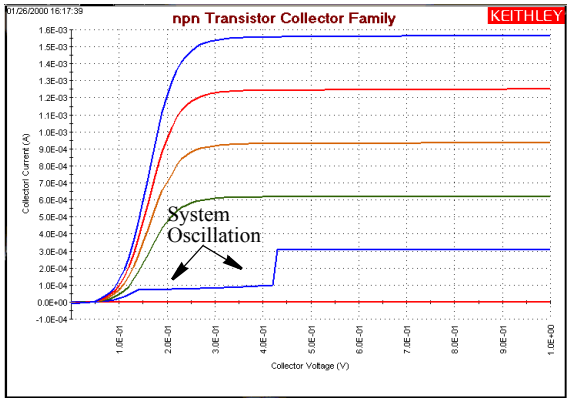
Using two or more SMUs to test an active device, such as a field-effect transistor (FET) or bipolar junction transistor (BJT), can aggravate system stability. Figure 5-14A shows an example of BJT characterization curves determined under stable conditions. Figure 5-14B shows an example of what can happen to a BJT characterization curve when the system oscillates.

Figure 5-14
Effects of oscillation on test data

A. Without Oscillation



B. With Oscillation



In general, oscillations can be classified in two categories: high-frequency oscillations (100 kHz through 200MHz), and low frequency-oscillations (below 100 kHz). For solutions to both types of oscillation, refer to Eliminating oscillations, below.

Eliminating oscillations

The measures needed to eliminate oscillations depend on whether the oscillations are high frequency or low frequency oscillations. The next two subsections treat these situations separately.

Eliminating high-frequency oscillations

One or more of the following remedies may help to eliminate high frequency oscillations; the remedies are listed in order of preference:

- Mount the preamps as close to the DUT as possible.
- Connect the COMMONs (outer shields) of all cables together at the DUT.
- Use loss ferrite beads or 100 Ω resistors in series with the DUT leads.

- Disconnect the ground link between GNDU COMMON and chassis ground on the rear panel of the mainframe. Connect the cable shields to the prober chassis.
- Add a high quality capacitor between the base and emitter of a bipolar junction transistor (BJT) or between the gate and source of an FET. Use a 100 pF to 1000 pF capacitor (Keithley Instruments part number C-138-100 pF).

Eliminating low frequency oscillations

Oscillations at low frequencies (DC to 100 kHz) occur when the gain of a transistor under test interacts with the output impedances of the connected SMUs. The following ratios of impedance (Z) determine the gains of the transistors:

- For a FET, $Z_{\text{Drain SMU}} / Z_{\text{Source SMU}}$
- For a BJT (bipolar junction transistor), $Z_{\text{Collector SMU}} / Z_{\text{Emitter SMU}}$

A SMU measures current via the voltage drop across a resistance, which is in series with the DUT. This series resistance is high for low current ranges and low for high current ranges. Therefore, for two SMUs connected to the transistor BJT collector and emitter terminals, or FET source and drain terminals, a large current-range difference (oscillation) results in the following:

- A large series-resistance difference
- A large impedance ratio between the two series resistances connected to the transistor
- A large circuit gain (potentially, the maximum, intrinsic transistor gain)
- A potentially unstable circuit

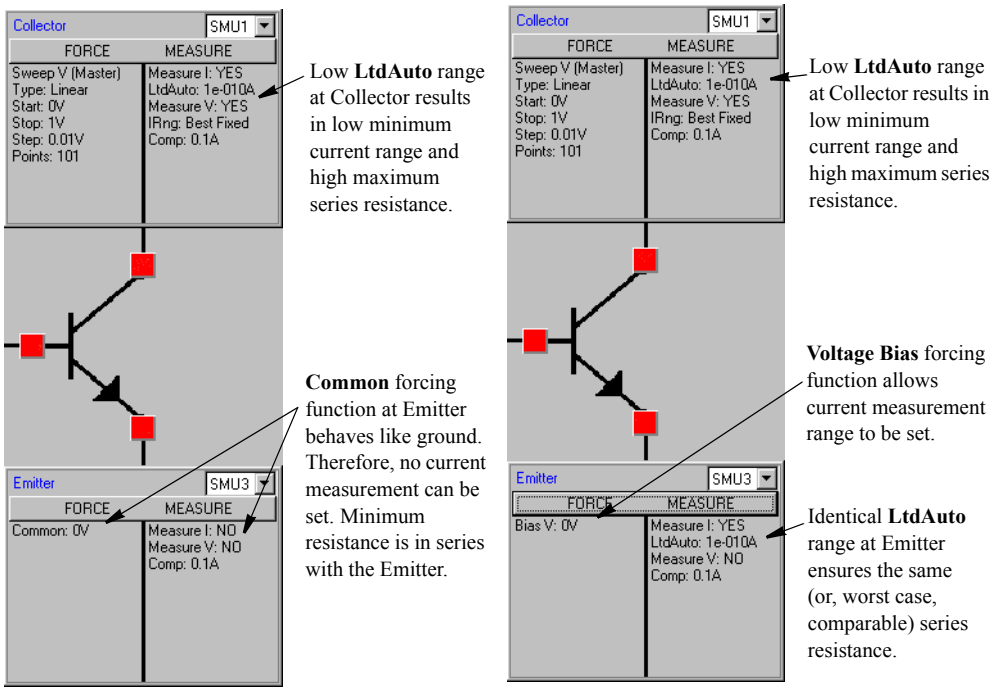
To avoid oscillations, try the following:

- For an FET:
  - Set (Drain-SMU current measure range) = (Source-SMU current measure range)
  - If necessary, set both SMUs to autorange.
  - Do not configure the source SMU for the **Common** forcing function, which prevents you from configuring a current measurement range for the source SMU and inevitably results in 1) a lower impedance than at the drain SMU; 2) a potentially high gain; and 3) an increased likelihood of low-frequency oscillation. Instead, configure the source SMU for the **Voltage-Bias** forcing function and set it to 0 V. This allows you to configure the current measurement range.
- For a BJT:
  - Set (Collector-SMU current measure range) = (Emitter-SMU current measure range)
  - If necessary, set both SMUs to autorange.
  - Do not configure the emitter SMU for the **Common** forcing function, which prevents you from configuring a current measurement range for the emitter SMU and inevitably results in 1) a lower impedance than at the collector SMU; 2) a potentially high gain; and 3) an increased likelihood of low-frequency oscillation. Instead, configure the emitter SMU for the **Voltage-Bias** forcing function and set it to 0 V. This allows you to configure the current measurement range.

NOTE *Both Drain/Collector and Source/Emitter (SMU) must be set to measure current if they are set to auto-range.*

Figure 5-15 contrasts undesirable and desirable KITE configurations for an Interactive Test Module (ITM) that is used to test a BJT.

Figure 5-15
Undesirable and desirable current measurement configurations for a BJT



NOTE *For instructions on configuring KITE ITMs, refer to Configuring the Project Plan ITMs in Section 6.*

Low current measurements

Low-current measurements made with a SMU or preamp are subject to a number of error sources that can have a serious impact on measurement accuracy. The following paragraphs discuss these low current measurement considerations.

Leakage currents

Sources of leakage currents

Leakage currents are generated by high-resistance paths between the measurement circuit and nearby voltage sources. These currents can considerably degrade the accuracy of low-current measurements.

Cable leakage currents

Typically, insulation resistance between conductors in the type of triax cables supplied with the SMUs and preamps is approximately 1 P Ω (10^{15} Ω). If the cables were used in an unguarded configuration, leakage current would flow through the cable insulation, affecting the measurement. Properly connecting the triax cables to the SMU or preamp automatically drives the inner cable shield at guard potential, minimizing the effects of cable leakage currents. See Guarding near the beginning of this section for details.

Reducing leakage currents

Several methods to reduce leakage currents include:

- Use good quality insulators, such as Teflon or polyethylene, in the test fixture.
- Reduce the humidity of the test environment. Insulators and even the test circuit itself may absorb water, causing spurious currents to be generated.
- Use guarding in the test fixture to isolate the high-impedance nodes from leakage current due to voltage sources. See Test fixture guarding earlier in this section for details.

Generated currents

Any extraneous generated currents in the test system will add to the desired current, causing errors. Currents can be internally generated, as in the case of preamp input offset current, or they can come from external sources such as insulators and cables. The following paragraphs discuss the various types of generated currents. Table 5-1 summarizes the typical ranges of a number of generated currents discussed in this section.

Table 5-1
Typical generated currents

Effect	Generated current range
Triboelectric	1 fA to 10 nA
Mechanical stress (Teflon)	1 fA to 1 pA
Mechanical stress (ceramics)	100 aA to 100 fA
Clean epoxy circuit board	100 fA
Dirty epoxy circuit board	100 pA

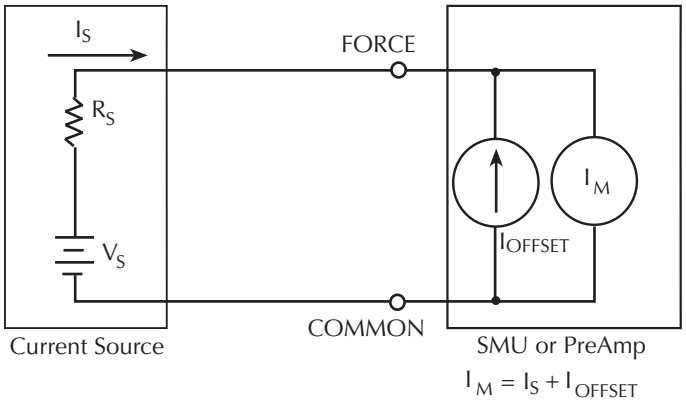
Offset currents

The preamp has a small current, known as the input offset current, that flows at all times. As shown in Figure 5-16A, the input offset current adds to the measured current so that the SMU measures the sum of the two currents. Note that input offset current can be nulled by performing system calibration, as discussed in Configuring the Project Plan ITMs in Section 6.

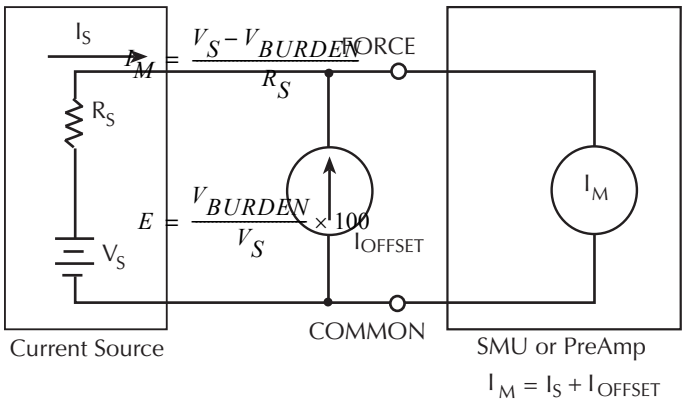
NOTE *The meaning of the word "nulled" in this context is to bring the offset current down to within specifications.*

Offset currents can also be generated externally from such sources as triboelectric and piezoelectric effects (discussed below). As shown in Figure 5-16B, the external offset current also adds to the source current, and the SMU again measures the sum of the two. These external offset currents can be suppressed manually by subtracting them via the KITE Formulator or the KITE **Calc** worksheet (for more information about the Formulator and the **Calc** worksheet, refer to Analyzing test data using the Formulator, and Displaying and analyzing data using the Sheet tab in Section 6).

Figure 5-16
Offset currents



A. Input Offset Current



B. External Offset Current

Triboelectric effects

Triboelectric currents are generated by charges created by friction between a conductor and an insulator. Here, free electrons rub off the conductor and create a charge imbalance that causes the current flow.

The triax cables supplied with the SMU and preamp greatly reduce this effect by using graphite-impregnated insulation underneath the outer shield. The graphite provides lubrication and a conducting cylinder to equalize and minimize charges generated by frictional effects of cable movement. However, even this type of triax cable creates some noise when subjected to vibration and expansion or contraction. Therefore, all connections should be kept short, away from temperature changes (which would create thermal expansion forces), and supported by taping or wiring the cable to a non-vibrating surface such as a wall, bench, or rigid structure.

Other solutions to movement and vibration problems include:

- Remove or mechanically decouple vibration sources such as motors, pumps, and other electromechanical devices.
- Securely mount or tie down electronic components, wires, and cables.
- Mount the preamps as close as possible to the DUT.

NOTE *A temporary triboelectric current is generated when a triax cable is first connected. This current is typically tens or hundreds of femtoamperes and can last as long as 5 to 10 minutes.*

Piezoelectric and stored charge effects

Piezoelectric currents are generated when mechanical stress is applied to certain crystalline materials used for insulated terminals and interconnecting hardware. In some plastics, pockets of stored charge cause the material to behave in a manner similar to piezoelectric materials.

To minimize the current due to this effect, remove mechanical stresses from the insulator, and use insulating materials such as polyethylene that have minimal piezoelectric and stored charge effects.

Contamination and humidity

Error currents also arise from electrochemical effects when ionic chemicals create weak batteries between two conductors on a circuit board. For example, commonly-used epoxy-printed circuit boards, when not thoroughly cleaned of etching solution, flux, or other contamination, can generate currents of a few nanoamps between conductors.

Insulation resistance can be dramatically reduced by high humidity or ionic contamination. High-humidity conditions occur with condensation or water absorption, while ionic contamination may be the result of body oils, salts, or solder flux.

To avoid the effects of contamination and humidity, select insulators that resist water absorption (such as Teflon), and keep humidity to <50% RH. Also be sure that all insulators are kept clean and free of contamination. If insulators become contaminated, clean them thoroughly with a pure solvent such as methanol.

Dielectric absorption

Dielectric absorption in an insulator can occur when a voltage across that insulator causes positive and negative charges within the insulator to polarize. When the voltage is removed, the separated charges generate a decaying current through circuits connected to the insulator as they recombine.

To minimize the effects of dielectric absorption on current measurements, avoid applying voltages greater than a few volts to insulators being used for sensitive current measurements. In cases where this practice is unavoidable, it may take minutes (or even hours in some cases) for the current caused by dielectric absorption to dissipate.

Voltage burden

As shown in Figure 5-17, the SMU or preamp ammeter may be represented by an ideal ammeter (I_M), with zero internal resistance, in series with a resistance (R_M). When a current source is connected to the input of the ammeter, the current is reduced from what it would be with the ideal resistance meter (R_M= 0 Ω). This reduction is caused by the resistance (R_M), which creates an additional voltage drop called the voltage burden (V_{BURDEN}), which reduces the measured current from its theoretical value as follows:

$$I_M = \frac{V_S - V_{BURDEN}}{R_S}$$

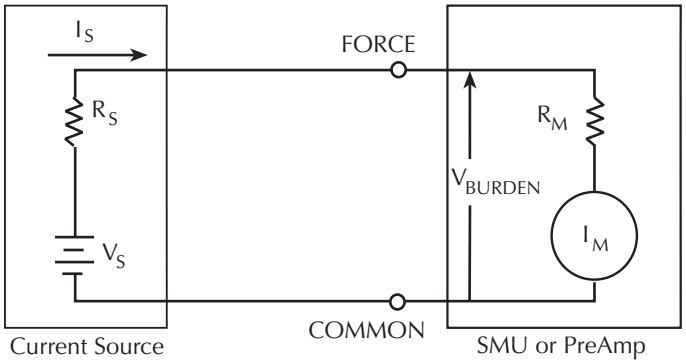
The percent error (E) in the measured reading due to voltage burden is:

$$E = \frac{V_{BURDEN}}{V_S} \times 100$$

If the voltage burden is 0 V, the percent error is zero.

NOTE *Voltage burden for the SMUs is less than or equal to the offset specifications of the source voltage. See Hardware features and capabilities in Section 1 for details on offset specifications.*

Figure 5-17
Effects of voltage burden



Noise and source impedance

Noise can seriously affect sensitive current measurements. The following paragraphs discuss how source resistance and source capacitance affect noise performance.

Source resistance

The source resistance of the DUT will affect the noise performance of the SMU or preamp. As the source resistance decreases, the current noise increases. Because decreasing the source resistance can have a detrimental effect on noise performance, there are usually minimum recommended source resistance values based on measurement range. Table 5-2 summarizes minimum recommended source resistance values for various measurement ranges.

Table 5-2
Minimum recommended source resistance values

Range	Minimum recommended source resistance
1 pA to 100 pA	1 G Ω to 100 G Ω
1 nA to 100 mA	1 M Ω to 100M Ω
1 μA to 100 μA	1 k Ω to 100 k Ω
1 mA to 100 mA	1 Ω to 100 Ω

Source capacitance

DUT source capacitance will also affect the noise performance of the preamp. In general, as source capacitance increases, the noise gain also increases. Although there is a limit to the maximum source capacitance value, you can usually measure at higher source capacitance values by connecting a resistor in series with the source. Note, however, that doing so will increase the voltage burden. For example, the source resistance values listed in Table 5-2 will result in a voltage burden between 1 mV and 1V.

Cable capacitance

Without guarding, the effects of cable capacitance would adversely affect the settling time when sourcing current. The rise time of the source depends on the total shunt capacitance seen at its output. For a high-impedance load, even a small amount of cable capacitance can result in long rise times. For example, cable capacitance of 100 pF and a load resistance of 1 G Ω will result in an RC time constant of approximately 100 msec. Guarding drastically reduces cable capacitance, resulting in much faster rise times. With FORCE and GUARD at virtually the same potential, the cable capacitance cannot charge, and rise time is not affected (refer to Guarding earlier in this section).

When sourcing voltage, the rise time due to cable capacitance is usually insignificant. Because the voltage source is low impedance (<1 Ω), the RC time constant of 10^{-10} seconds 1 Ω × 100 pF is negligible.

Performance of an integrated semiconductor test system

When performing a semiconductor I-V measurement, there will always be a speed-noise trade-off. Even with given measurement settings, changing the system configuration (such as cable length or adding a switch matrix) will change the measurement results. The Model 4200-SCS has four settings to allow optimal I-V measurements. There are three fixed settings: fast, normal, and quiet. In addition, there is a custom setting to allow the measurement parameters to be customized.

To achieve a low-noise measurement, the quiet setting is recommended. The trade-off is that measurement speed will be lower in comparison to the fast and normal settings. To make a fast measurement, the fast setting can be selected, though the noise will be higher. Typically, the normal setting is used to balance the speed and low-noise requirements. To further fine-tune the measurement, the custom setting can be used.

The fast/normal/quiet settings are tuned to the Model 4200-SCS for a standard length of cables connected to the DUT. In general, this should be sufficient to make good measurements. However, when extra long cables and/or a switch matrix are used in the system, these settings may not be adequate. A typical phenomenon will be the appearance of a glitch or offset error. The magnitude of the error increases if the fast setting is used to make the measurement. This is caused by insufficient settling time for the system. With added load or capacitance (cables or matrix relays), it will take longer to let transient effects settle. Using the measurement parameters optimized for short cables only may result in an erroneous measurement.

The best way to minimize this effect is to allow extra settling time. The normal or quiet settings should improve the measurement result. Custom can also be used to fine-tune the measurement settings; this may be a trial and error process. Various combinations of parameters can be used to achieve the best results. In general, longer cables or slower settling of switch relays will require a larger delay factor.

Interference

Various forms of interference that can degrade measurement integrity include:

- Electrostatic interference
- Radio frequency interference
- Ground loops

Electrostatic interference

Electrostatic interference occurs when an electrically charged object is brought near an uncharged object, thus inducing a charge on the previously uncharged object. Usually the effects of such electrostatic action are not noticeable because low impedance levels allow the induced charge to dissipate quickly. However, the high impedance levels of many SMU or preamp measurements do not allow these charges to decay rapidly, and erroneous or unstable readings may be caused in the following ways:

- DC electrostatic fields can cause undetected errors or noise in the reading.
- AC electrostatic fields can cause errors by driving the amplifier into saturation, or through rectification that produces DC errors.

Electrostatic interference is first recognizable when hand or body movements near the DUT cause fluctuations in the reading. Pick-up from AC fields can also be detected by observing the output on an oscilloscope.

Means of minimizing electrostatic interference include:

- **Shielding**: Possibilities include: a shielded room, a shielded booth, shielding the sensitive circuit (test fixture), and of course using shielded cable. The shield should usually be connected to a solid connector that is connected to signal COMMON. Note, however, that shielding can increase capacitance, possibly slowing down response time unless guarding is used within the test fixture.
- **Reduction of electrostatic fields**: Moving power lines or other sources away from the DUT reduces the amount of electrostatic interference induced into the test circuit.

Radio frequency interference

Radio Frequency Interference (RFI) is a general term frequently used to describe electromagnetic interference over a wide range of frequencies across the spectrum. RFI can be especially troublesome at low signal levels, but it may also affect higher level measurements in extreme cases.

RFI can be caused by steady-state sources such as TV or radio broadcast signals, or it can result from impulse sources, as in the case of arcing in high voltage environments. In either case, the effect on measurement performance can be considerable if enough of the unwanted signal is present. The effects of RFI can often be seen as an unusually large offset, or, in the case of impulse sources, sudden, erratic variations in readings.

Different methods can be used to minimize the effects of RFI. The most obvious method is to keep DUT as far away from the RFI source as possible. Shielding the test equipment, DUT, and test cables often reduces RFI to an acceptable level. In extreme cases, a specially-constructed screen room may be necessary to sufficiently attenuate the troublesome signal.

Ground loops and other SMU grounding considerations

Ground loops, which occur when more than one point in a test system is connected to earth ground, can create error signals that cause erratic or erroneous performance. The configuration shown in Figure 5-18 shows a ground loop that is created by connecting both Model 4200 signal COMMON and DUT LO to earth ground. A large ground current flowing in the loop will encounter small resistances, either in the conductors, or at the connecting points. This small resistance results in voltage drops that can affect performance.

To prevent ground loops, the test system should be connected to ground at only a single point. If it is not possible to remove the DUT ground, the ground link between the GNDU COMMON terminal and chassis ground should be removed, as shown in Figure 5-19. Note, however, that removing the COMMON-to-chassis link may result in oscillations (refer to Making stable measurements earlier in this section).

Figure 5-18
Ground loops

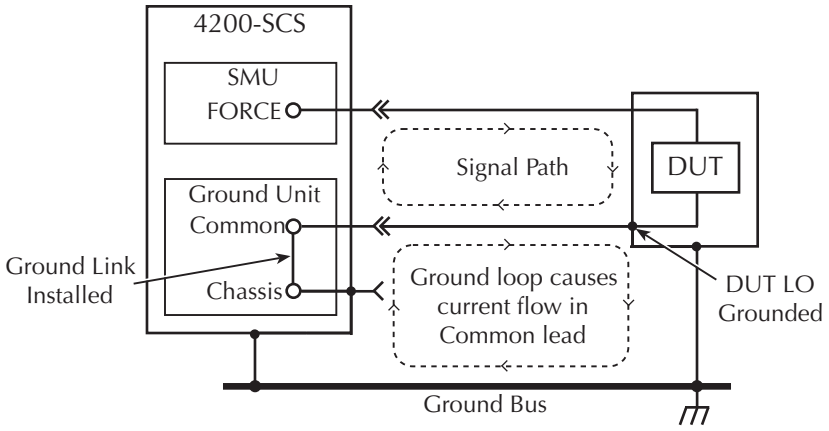
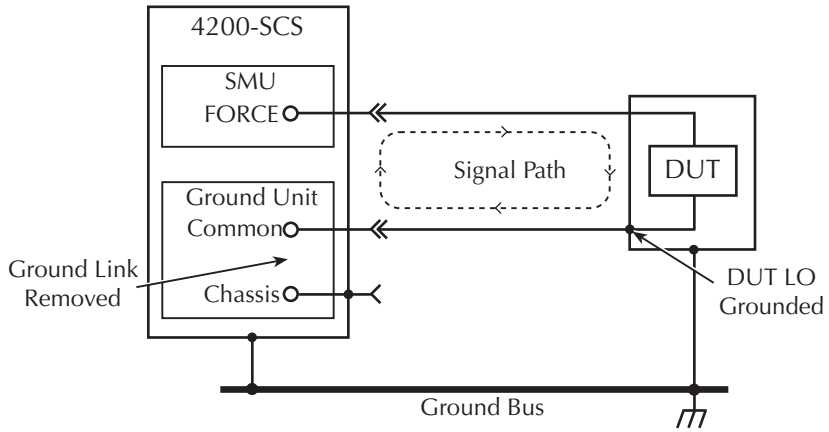


Figure 5-19
Eliminating ground loops



Section 6

Keithley Interactive Test Environment (KITE)

In this section:

Introduction

This section of the manual explains and illustrates the characteristics and application of the Keithley Interactive Test Environment (KITE). KITE is the main software component of the KTE Interactive software tool set. KITE is the primary user interface for the Keithley Instruments Model 4200-SCS Semiconductor Characterization System.

KITE is a versatile tool that can be used to run project tests that are provided with the Model 4200-SCS (see KITE projects). You can also use KITE to modify existing projects and/or tests or create new ones. KITE facilitates interactive characterization of an individual parametric test device or automated testing of an entire semiconductor wafer.

Two additional KTE Interactive software tools augment the capabilities of KITE, as follows:

- The Keithley User Library Tool (KULT) is used to create test modules using the C programming language. KITE can then execute these test modules.
- The Keithley CONfiguration utility (KCON) is used to manage the configuration and interconnections between all of the test system components that are controlled by KITE.

A fourth KTE Interactive software tool, the Keithley External Control Interface (KXCI) allows the Model 4200-SCS to be controlled remotely by an external GPIB controller.

NOTE *KXCI and KITE are mutually exclusive software tools. That is, KXCI and KITE cannot run simultaneously.*

Beginning with KTE Interactive 6.0, two optional KTE Interactive tools have been added:

- The Keithley Pulse tool (KPulse) is a virtual front panel software application used to control the optional Keithley pulse cards. A pulse card is a dual-channel pulse generator that is integrated inside the Model 4200-SCS mainframe.
- The Keithley Scope tool (KScope) is a virtual front panel software application used to control the optional scope card. The scope card is a dual-channel Digital Storage Oscilloscope that is integrated inside the Model 4200-SCS mainframe.

NOTE *KScope and KPulse can be launched at the same time as KITE. However, KScope, KPulse and KITE cannot communicate with the hardware simultaneously.*

KITE projects

A KITE project is a collection of Individual tests that were created in KITE. A project can consist of a single test or multiple tests. Tests in a project can be performed on a single device or multiple devices. Kite projects are organized in project folders. A project folder contains one or more projects. For example, the project folder named “Memory” contains the projects for testing flash memory. Table 6-1 lists and summarizes the Kite projects.

NOTE *Following Table 6-1, details on using KITE starts with an Overview of KITE.*

.

Table 6-1
KITE projects

Project Folder	Project	Description
_BJT	BJT-Default	Model 4200-SMU performs common BJT tests (vce-ic, gummel, saturation voltage).
_CMOS	CMOS-Default	Model 4200-SMU performs common BJT tests (vce-ic, gummel, saturation voltage).
_CV	CVU-BJT	Model 4200-CVU measures C-t (0V) between terminals of a BJT. See Section 15.
	CVU_Capacitor	Model 4200-CVU measures C-V and C-f on a capacitor. See Section 15.
	CVU_highV	Uses a Model 4200-CVU, Model 4200-SMU, and Model 4205-RBT remote bias tees to perform high voltage C-V measurements. Tests are provided for a zener diode, MOScap, capacitor, and schottky diode. See Appliction Note 2972.
	CVU_InterconnectCap	Model 4200-CVU measures interconnect capacitance. See Section 15.
	CVU_ivcvswitch	Model 4200-708 Matrix switches Model 4200-SMU and Model 4200-CVU to measure I-V and C-V on a capacitor. See Section 15.
	CVU_Lifetime	Model 4200-CVU measures C-V, C-t, and generates Zerbst plot on a MOScap. See Section 15.
	CVU_MobileIon	Model 4200-CVU is used to determine mobile ion charge concentration using the bias temperature stress (BTS) method. See Section 15.
	CVU_MOScap	Model 4200-CVU performs a C-V sweep and generates a doping profile on a MOScap. Parameter extractions including C_{OX} t_{OX}, V_{TH}, V_{FB}, etc. See Section 15 and Application Note 2896.
	CVU_MOSFET	Model 4200-CVU performs a C-V sweep and doping profile on a MOSFET. Includes parameter extractions. See Section 15.
	CVU_nanowire	Model 4200-CVU performs a C-V sweep on a nanowire. See Section 15.
	CVU_Pnjunction	Model 4200-CVU performs a C-V sweep and doping profile on a PN junction. Includes parameter extractions. See Section 15.
	CVU_Pvcell	Model 4200-CVU and Model 4200-SMU perform C-V and I-V measurements on solar cells. Includes parameter extractions. See Appliction Note 2973. See Application Note 2876.
	ivcvswitch	Model 590 CV Meter, Model 4200-SMU, and a HP-4284A LCR Meter are switched with a Model 707A/708A to make C-V and I-V measurements on a MOScap. See Appendix C and Appendix D.
	lifetime	Controls the Model 82 C-V System to extract the minority carrier recombination for lifetime measurements (Zerbst plot). See Generation velocity and generation lifetime (Zerbst plot) in Appendix E.

Table 6-1 (continued)
KITE projects

Project Folder	Project	Description
_CV (continued)	qscv	Two Model 4200-SMUs used to measure quasistatic CV. See Application Note 2973.
	simcv	Controls Model 82 to make simultaneous C-V measurements. Includes formulas for parameter extractions. See Section 15.
	stvs	Controls Model 82 to determine mobile ion charge concentration using the triangular voltage sweep method. See Mobile ion charge concentration in Appendix E.
	VLF_CV_Examples	Uses two SMUs with preamps to provide C-V testing from 10 mHz to 10 Hz. See the VLF CV App Note link on the Applications page of the 4200 Complete Reference.
_Demo	Demo-ALL	This is a collection of the vast majority of the the tests that are provided by the Model 4200.
	Demo-Default	Performs a fast MOSTFET family of curves.
	Demo-PulseIV	PIV-A demo (see How to perform a Pulsed I-V test on my device in Section 3 of the User's Manual.
	Demo-QPulseIV	PIV-Q demo (see How to perform a Pulsed I-V test on my device in Section 3 of the User's Manual.
_Memory	FlashDisturb-NAND	Uses four pulse generators and two Model 4200-SMUs to pulse stress a device and measure Vt on an adjacent device. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	FlashDisturb-NOR	Uses four pulse generators and two Model 4200-SMUs to pulse stress a device and measure Vt on an adjacent device. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	FlashDisturb-Switch	Uses four pulse generators, two Model 4200-SMUs, and a Model 707A (for switching) to pulse stress a device and measure Vt on an adjacent device. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	FlashEndurance-NAND	Uses four pulse generators and two Model 4200-SMUs to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	FlashEndurance-NOR	Uses four pulse generators and two Model 4200-SMUs to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	FlashEndurance-Switch	Uses four pulse generators, two Model 4200-SMUs, and a Model 707A (for switching) to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	Flash-NAND	Uses four pulse generators and two Model 4200-SMUs to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.

Table 6-1 (continued)
KITE projects

Project Folder	Project	Description
_Memory (continued)	Flash-NOR	Uses four pulse generators and two Model 4200-SMUs to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	Flash-Switch	Uses four pulse generators, two Model 4200-SMUs, and a Model 707A (for switching) to perform program/erase cycles and then measure Vt. See How to perform a flash memory test on my device in Section 3 of the User's Manual.
	PMU-Flash-NAND	This project demonstrates the capabilities of the Model 4225-PMU for FLASH memory testing (see PMU-Flash-NAND project).
	NVM_Examples	Uses 4225-PMU, 4225-RPM and SMUs to characterize NAND flash, phase change memory, and ferroelectric memory. See the NVM Application Note link on the Applications page of the 4200 Complete Reference.
_Miscellaneous	ivswitch	Same as default project but uses Model 708 switching. See Appendix B.
	LowCurrent	Uses Model 4200-SMU to measure offset current and then graphs offset current vs. time. See Application Note 2959.
	probesites	Uses a switch matrix, prober control, and Model 4200-SMUs to generate a family of curves (drain current vs. drain voltage) on wafer sites. See Appendix G.
	probesubsites	Uses a switch matrix, prober control, and Model 4200-SMUs to generate a family of curves (drain current vs. drain voltage) on wafer subsites. See Appendix G.
_Nanotech	NanoDevices	Multiple DC I-V measurements on a wide variety of nanodevices.
	CNTFET	Performs optimal DC I-V, pulsed I-V, and C-V measurements on carbon nanotube (CNT) FETs. See CNT FET Application Note link on the Applications page of the 4200 Complete Reference.
_Pulse	chargepumping	Uses a pulse generator and two Model 4200-SMUs to make chargepumping measurements using several methods (see).
	chargetrapping	Uses a pulse generator to perform a charge trapping test. See Section 3 of the User's Manual and Application Note 2518.
	ivpgswitch	Performs automated measurements on a MOSFET using the Model 708 switch matrix, HP8110 pulse generator, and Model 4200-SMUs. See Appendix F.
	ivpgswitch_340x	Same as the ivpgswitch project but uses a Model 340x pulse generator instead of the HP8110 pulse generator.
	PMU-DUT-Examples	This project contains example test modules to test a MOSFET using the Model 4225-PMU (see PMU-DUT-Examples project).

Table 6-1 (continued)
KITE projects

Project Folder	Project	Description
_Pulse (continued)	PMU-MOSFET	This project uses the Model 4225-PMU to perform 3-terminal ultra fast I-V and DC tests on a MOSFET (see PMU-MOSFET project).
	PMU-Switch	This project provides examples for switching between a Models 4225 PMU, Model 4200-SMU, and Model 4200-CVU to the device under test (DUT) (see PMU-Switch project).
	PulseIV-Complete	Uses the Model 4200-PIV-A package to perform Pulse I-V tests. See How to perform a Pulsed I-V test on my device in Section 3 of the User's Manual.
	QPulseIV-Complete	Uses the Model 4200-PIV-A package to perform Pulse I-V tests. See How to perform a Pulsed I-V test on my device in Section 3 of the User's Manual and PA-956 (4200-PIV-Q Applications Note).
_Reliability	EM_const_1	Electromigration test of a single device using a Model 4200-SMU. Uses subsite cycling. See Appendix M. For subsite cycling, see Subsite cycling (in this section) and How to perform AC stress for wafer level reliability (WLR) in Section 3 of the User's Manual.
	HCI_1_DUT	Hot carrier injection degradation of a single device using Model 4200-SMUs. Uses subsite cycling. See Appendix M. For subsite cycling, see Subsite cycling (in this section) and How to perform AC stress for wafer level reliability (WLR) in Section 3 of the User's Manual.
_Reliability	HCI_4_DUT	Same as the HCI_1_DUT project except it tests four devices using a switch matrix for connections. Uses subsite cycling. See Appendix M. For subsite cycling, see Subsite cycling (in this section) and How to perform AC stress for wafer level reliability (WLR) in Section 3 of the User's Manual.
	HCI_PULSE	Hot carrier injection degradation of a single device using Model 4200-SMUs, a pulse generator and Model 707A for switching. See How to perform AC stress for wafer level reliability (WLR) in Section 3 of the User's Manual.
	NBTI_1_DUT	Performs Negative Bias Temperature Instability (NBTI) on a single 4-terminal MOSFET. Uses Model 4200-SMUs, a Model 707A for switching, and provides hot chuck control. See Appendix M. For subsite cycling, see Subsite cycling (in this section) and How to perform AC stress for wafer level reliability (WLR) in Section 3 of the User's Manual.
	Qbd	Uses a Model 4200-SMU to Perform Ramp V and Ramp J tests. See V-ramp and J-ramp tests in Appendix M.
_Resistivity	FourPtProbe	Measures resistivity using 4-point collinear probe method with either three or four Model 4200-SMUs. See Application Note 2475.
	vdp_resistivity	Measures vdp resistivity using four Model 4200-SMUs. The resistivity data appears in the subsite. See Application Note 2475.

Table 6-1 (continued)
KITE projects

Project Folder	Project	Description
_Solar	SolarCell	Uses Model 4200-SMUs and a Model 4210-CVU for basic solar cell testing. There are tests for I-V sweep, 4-point probe resistivity, van der Pauw resistivity, C-V sweep, and DLCP. See Application Note 3026.
default	default	Common DC I-V and C-V tests for MOSFETs, BJTs, resistors, diodes, and capacitors. Table 16-17 summarizes the default tests for the Model 4225-PMU.

Overview of KITE

This subsection overviews the primary features of KITE. These features allow you to create, execute, and evaluate tests and complex test sequences, interactively and without programming. This subsection also overviews use of an essential companion tool, KULT, that allows you to create libraries of specialized user modules that run in KITE (KULT is discussed in detail in Keithley User Library Tool (KULT) in Section 8).

KITE interface

KITE interface consists of a variety of graphical user interfaces (GUIs) that allow you to do the following:

- Interactively build and edit test and execution sequences using the Project Navigator.
- Configure off-the-shelf interactive test modules (ITMs) or create new, customized ITMs from off-the-shelf templates.
- Create user test modules (UTMs) from supplied or user-programmed C-code modules.
- Automatically execute tests and associated operations (switch matrix connections, prober movements, etc)., including:
  - A single test for one selected device (transistor, diode, resistor, capacitor, etc)..
  - A test sequence for one selected device.
  - Test sequences for multiple devices, for example, all of the devices contacted by a prober at a given touchdown, or subsite, location on a semiconductor wafer.
  - The test sequences of an entire Project Plan, which may include multiple prober touchdowns for a single semiconductor site (or die):
    - For one site
    - For multiple sites
- View test results, numerically and graphically.
- Analyze test results using built-in parameter extraction tools.
- View the analysis results, numerically, and graphically.

The KITE interface overviews the KITE interface. The next subsection, Project Navigator, describes the Project Navigator in more detail.

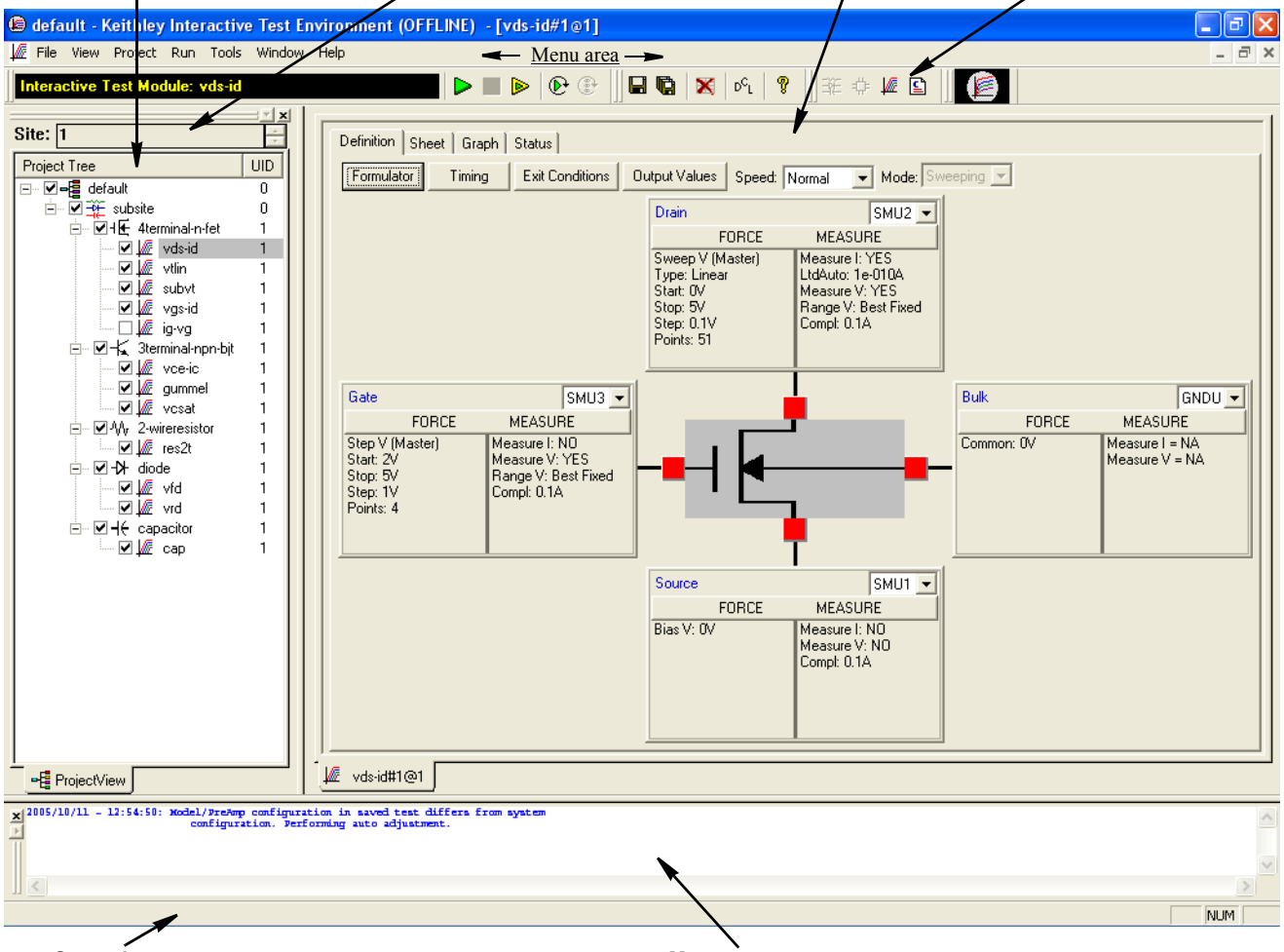
Figure 6-1
KITE interface overview

Project Navigator:
Where a Project Plan is assembled, edited, displayed, and executed (A Project Plan defines a series of tests, of various devices, at one or more locations). Double-clicks here lead to definition, configuration, and tool screens. A selection here defines the starting location when only part of the Project Plan is to be executed.

Site Navigator:
Displays the current site—typically a die on a semiconductor wafer—that is being evaluated by the Project Plan. Allows selection of the single site to be evaluated when only part of the Project Plan is to be executed.

KITE Workspace:
Displays the variety of screens, windows, tabs, message boxes, and so on. that are used: 1) to configure all Project-Plan components; 2) observe evaluation results; and 3) analyze evaluation results.

Toolbar Area:
Displays a variety of icons that can be used to: 1) start and stop all or part of a Project Plan; 2) verify Project Plan execution; 3) insert Project Plan components; 4) save and print Project Plan files; and 5) view KITE help.



Status bar:
Displays descriptions of menu and toolbar

Message area:
Displays KITE execution and error messages.

Project Navigator

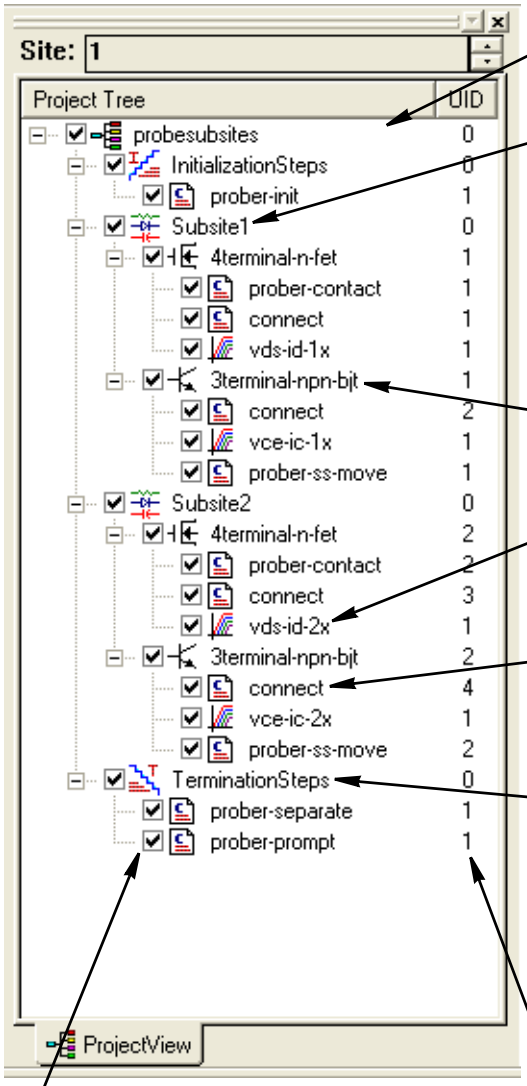
The Project Navigator is the primary interface for building, editing, and viewing a Project Plan, and for specifying and accessing each Project Plan component, as follows:

- Each Project Plan component may be added, sequentially or nonsequentially, using menu items or toolbar buttons.
- Single-clicking on a Navigator component selects it as one of the following:
  - A location where a new component may be added or an existing component may be deleted.
  - An individual test, device, or series of devices for which only part of the Project Plan may be executed.
- Double-clicking on a Navigator component opens access to configuration screens for the component and, as appropriate, test results, analysis tools, and status information.

Figure 6-2 describes typical Project Plan components that are displayed in a Project Navigator, using the **example** Project Plan for illustration. The next subsection, Interactive Test Modules (ITMs) and User Test Modules (UTMs), describes the ITM and UTM components in more detail.

For details about building a Project Plan using the Project Navigator, refer to Building, modifying, and deleting a Project Plan later in this section.

Figure 6-2
Project Navigator



Project Plan:
Defines and sequences all subsites tested, all devices tested, and all tests/operations to be performed at each site—which typically corresponds to one die on a wafer.

Subsite Plan:
Defines and sequences all devices to be tested and all tests to be performed at a given prober touchdown location. There are typically multiple subsites per site. Subsite cycling (up to 128 times) can be performed to accommodate stress testing. For the first cycle, the devices in the Subsite Plan are not stressed. For each subsequent stress cycle, the devices can be stressed with voltage or current for a specified period of time before they are tested. Each stress cycle can have a unique stress time, and each device can have unique device stress properties.

Device Plan:
Defines and sequences all tests for a specific device—a transistor, diode, resistor, and so on.—at a given subsite.

Interactive Test Module (ITM):
Completely defines a parametric test without programming, using a series of easily configured graphical user interfaces (GUIs). Provides for display of both raw data and analyzed data, numerically and graphically, in real time.

User Test Module (UTM):
Defines an operation—a special test, a setting of switch-matrix connections, a prober advance, an external instrument operation, and so on.— using a C-programmed user module that is connected to the UTM and is configured with user-supplied parameter values (Several UTMs may be associated with the with the same user module). Configuration is done using a simple graphical user interface. The user module that is connected to the UTM may be available in a Keithley-supplied library or may be created by the user with KULT. Provides display of both raw and analyzed test data, where applicable, numerically and graphically.

Initialization Steps (at top) and **Termination Steps:**
Define operations (UTMs only) that initialize test equipment at the beginning of a test session and process results or reset instrumentation at the end of an execution sequence. The initialization and termination steps are executed only once during a test session, even if the Project Plan is executed several times (for example, to evaluate multiple sites on a wafer).

Unique ID (UID) number:
A number assigned to each instance of a same-named project component. If there is only one instance of a Project Plan component of a given name (as in the Project Plan at left), each component has a UID of 1. However, if multiple components, for example, ITMs or UTMs, of a given name are inserted into a project, they are assigned multiple UIDs. As long as a component remains in the Project Plan, its UID never changes—even if a lower-numbered same-named component is deleted from the Project Plan (Note: if UID = 0 for a component, that component can occur only once in the Project Plan).

Project Navigator Checkboxes:

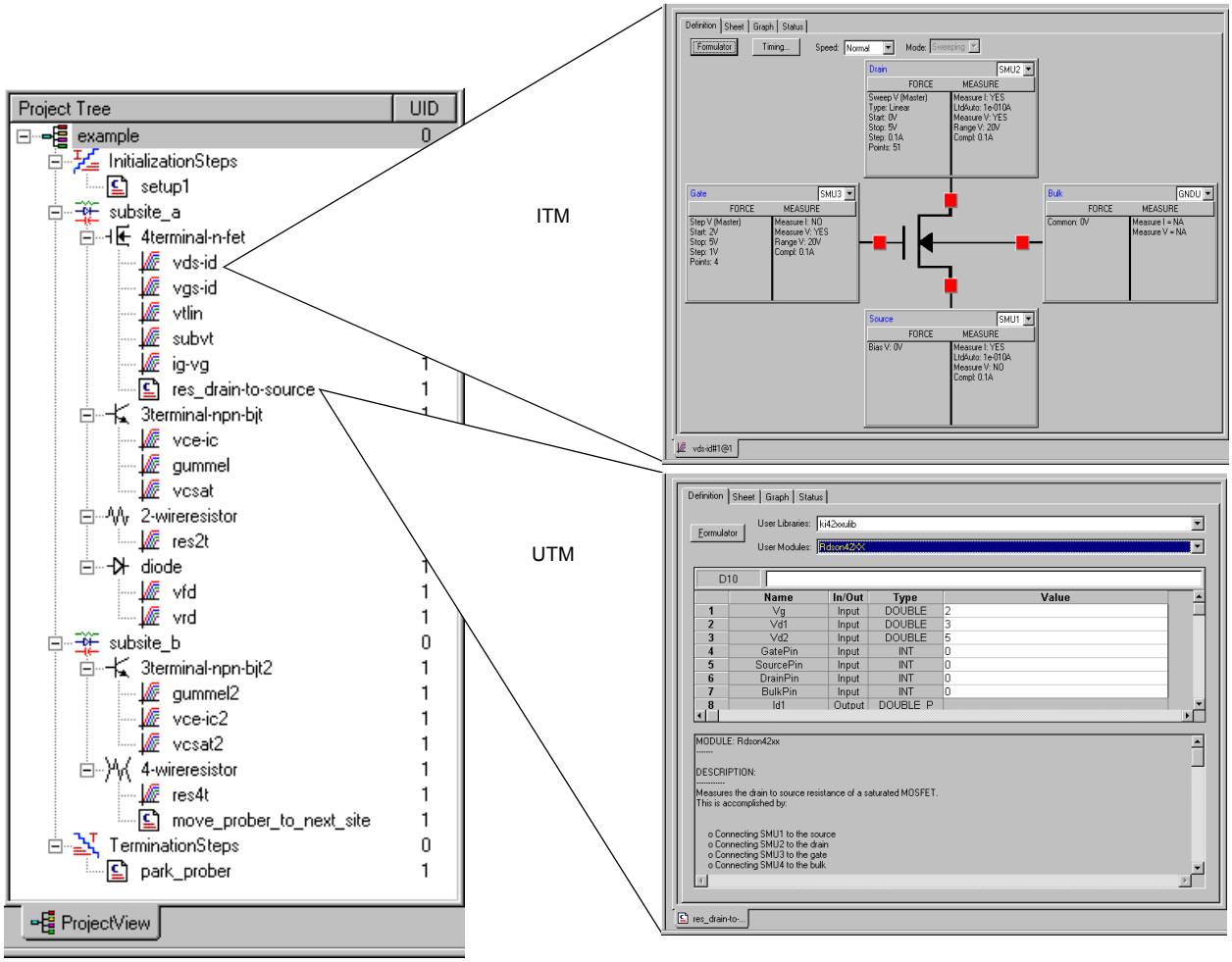
Project Navigator Checkboxes are used to enable or disable the Project Plan, Init/Term Steps, Subsite Plans, Device Plans, and individual tests (ITMs and UTMs). Project nodes that are checked will be executed when the Run, Append, or Cycle buttons are clicked. Project nodes that are unchecked will be skipped during any execution sequence.

Interactive Test Modules (ITMs) and User Test Modules (UTMs)

KITE tests and operations are performed through interactive test modules (ITMs) and user test modules (UTMs), as called out in Figure 6-2. Figure 6-3 relates configuration windows of the **vds-id** ITM and the **res_drain-to-source** UTM to their respective locations in the example KITE Project Plan.

These windows and some associated windows are examined in more detail in the next two subsections, Defining an ITM, and Defining a UTM.

Figure 6-3
ITMs and UTMs in the Project Navigator



The primary differences between ITMs and UTMs are summarized in Table 6-2.

Table 6-2
Primary differences between an ITM and a UTM

ITM	UTM
Is always configured using a series of systematic, interactive graphical user interfaces (GUIs), without programming.	Is created and configured by connecting the UTM name to a user module and entering or modifying the input parameter values.
Is flexible. Keithley Instruments provides default ITM configurations for most standard devices and tests; you may be able to perform many of your evaluations with minimal or no changes to the default parameters. However, you can create a new ITM, or customize any existing ITM, to perform a wide variety of static and dynamic evaluations. You can even create an ITM for a generic "n-terminal" device.	Is task-specific. However, you can modify the source code for a user module that is connected to a UTM and recompile it to create a new user module. Keithley Instruments provides the source code for most of the user modules that are shipped with the Model 4200-SCS. User modules are modified using KULT.
Performs exclusively tasks on internal 4200-SCS instrumentation.[1]	Performs tasks on internal Model 4200-SCS instrumentation or on any instrument that is connected to the Model 4200-SCS IEEE-488 bus or the Model 4200-SCS RS-232 port.
Is used exclusively for parametric testing.	May be used to perform almost any test-related task.
Generated data updates the **Data** worksheet[2] in real time, as the test executes.	Generated data updates the **Data** worksheet[1] after test execution is complete. Beginning with KTE Interactive v5.0, users have the ability to add function calls to new and existing user modules (UTMs) that provide real-time data and graphing capabilities (see "Enabling real time plotting for UTMs").

1. The Models 4205-PG2 and 4220-PGU pulse cards, and the scope card (Models 4200-SCP2 or 4200-SCP2HR) are not supported by ITMs at this time. Keithley Instruments does now offer the Model 4225-PMU pulse source and measure card which is supported by ITMs.
2. Refer to "Viewing ITM or UTM results numerically: The Sheet tab Data worksheet" later in this section.

The following subsections explains how to enable real time plotting for UTMs and illustrate some of the differences (and similarities) between ITMs and UTMs:

- Enabling real time plotting for UTMs
- Defining an ITM
- Defining a UTM
- Viewing ITM or UTM results numerically: The Sheet tab Data worksheet
- Viewing ITM or UTM results graphically: The Graph tab

Enabling real time plotting for UTMs

A limitation of UTMs in KITE software in previous software releases is that they lacked the capability of real time plotting. All data can only be plotted after the test is finished. This sometimes becomes inconvenient when the test takes a very long time. It is not easy to know what is going on during the test. Since the release of version 5.0, it has become possible to plot data in real time. The following explains how to enable real time plotting in UTM.

Beginning with KTE version 5.0, there were three new functions added in the existing Linear Parametric Test (LPT) library. Those are `PostDataDouble()`, `PostDataInt()`, and `PostDataString()` respectively. The protocols of the three functions are listed as follows:

```
PostDataDouble(char *, double *)
PostDataInt(char *, int *)
PostDataString(char *, char *)
```

These functions are used right after one measurement point is finished and data value is assigned to the corresponding output variable. They will transfer the data value from memory to the data sheet in the UTM and plot it on the graph. `PostDataDouble()` is used to transfer a double type data point from memory back to the data sheet. `PostDataInt()` is used to transfer an integer type data point from memory back to the data sheet. `PostDataString()` is used to transfer a string from memory back to the data sheet. Each function should be used according to type defined for the corresponding output parameter. For example, if an output parameter is defined as `double`, then `PostDataDouble()` function should be used to bring data back to the data sheet and then plot it in real time.

The first parameter in the three functions is the variable name, defined as char *. For example, if the output variable name is DrainI, then `DrainI` (with quotes) should be used in the first parameter. The second parameter is the value of the variable to be transferred. For example, if DrainI[10] should be transferred, then one should call `PostDataDouble("DrainI", DrainI[10])`.

When using the new functions to transfer data into the data sheet in real time, make sure the data is already located in the memory of the Model 4200-SCS. Sweep measurements are not suitable for real time transfer because data is not ready until sweep finishes. The following examples show how to enable real time plotting for a UTM.

1. I-V measurement using SMUs of a Model 4200-SCS

   As mentioned above, if real time plotting is desired, sweep type of measurement cannot be used. Therefore, functions that are sweep related, such as smeasx(), sintgx(), and sweepx(), cannot be used. Instead, one should form a for loop in the program to do point-by-point measurement. Here is one example. In this example, SMU1 is used to force voltage and then measure current. Programmed voltage and measured current are then outputted for plotting.

   Without real time plotting:

```
#include "keithley.h"
int IV(double startv, double stopv, int numpoint,
double * V, int Vsize, double * I, int Isize)
{
// error checking
if ((numpoint != Vsize) ||(numpoint != Isize))
return -1;
rtfary(V); // return force array of Voltage
sintgi(SMU1, I); // measure current
// setup sweep with 0.01 second sweep delay and
then trigger the measurement
sweepv(SMU1, startv,stopv,numpoint-1,0.01);
return (OK);
```

```
}
```

With real time plotting:

```
#include "keithley.h"
int IV(double startv, double stopv, int numpoint,
double * V, int Vsize, double * I, int Isize)
{
int index;
double stepv;
// error checking
if ((numpoint != Vsize) ||(numpoint != Isize))
return -1;
//calculate stepv
if (numpoint != 1)
stepv = (stopv - startv)/(numpoint -1);
// measurement loop
for (index = 0; index < numpoint; index ++)
{
// calculate voltage array
V[index] = startv + stepv*index;
forcev(SMU1, V[index]); // force voltage
intgi(SMU1,&I[index]); // measure current
// transfer V data
PostDataDouble("V", V[index]);
// transfer I data
PostDataDouble("I", I[index]);
}
return (OK);
}
```

2. Taking measurements using external instruments

   The same idea applies to measurements using external instruments. A trigger (sweep) data method cannot be used because real time plotting requires real time data. In the above method, the data will not be ready until the sweep is done. The following is an example of getting real time data from external instruments.

   Test sequence without real time plotting:

```
initialize instrument;
setup sweep;
setup trigger; trigger measurement;
serial poll the instrument until measurement is done;
retrieve data from instrument;
post measurement clean up;
done;
```

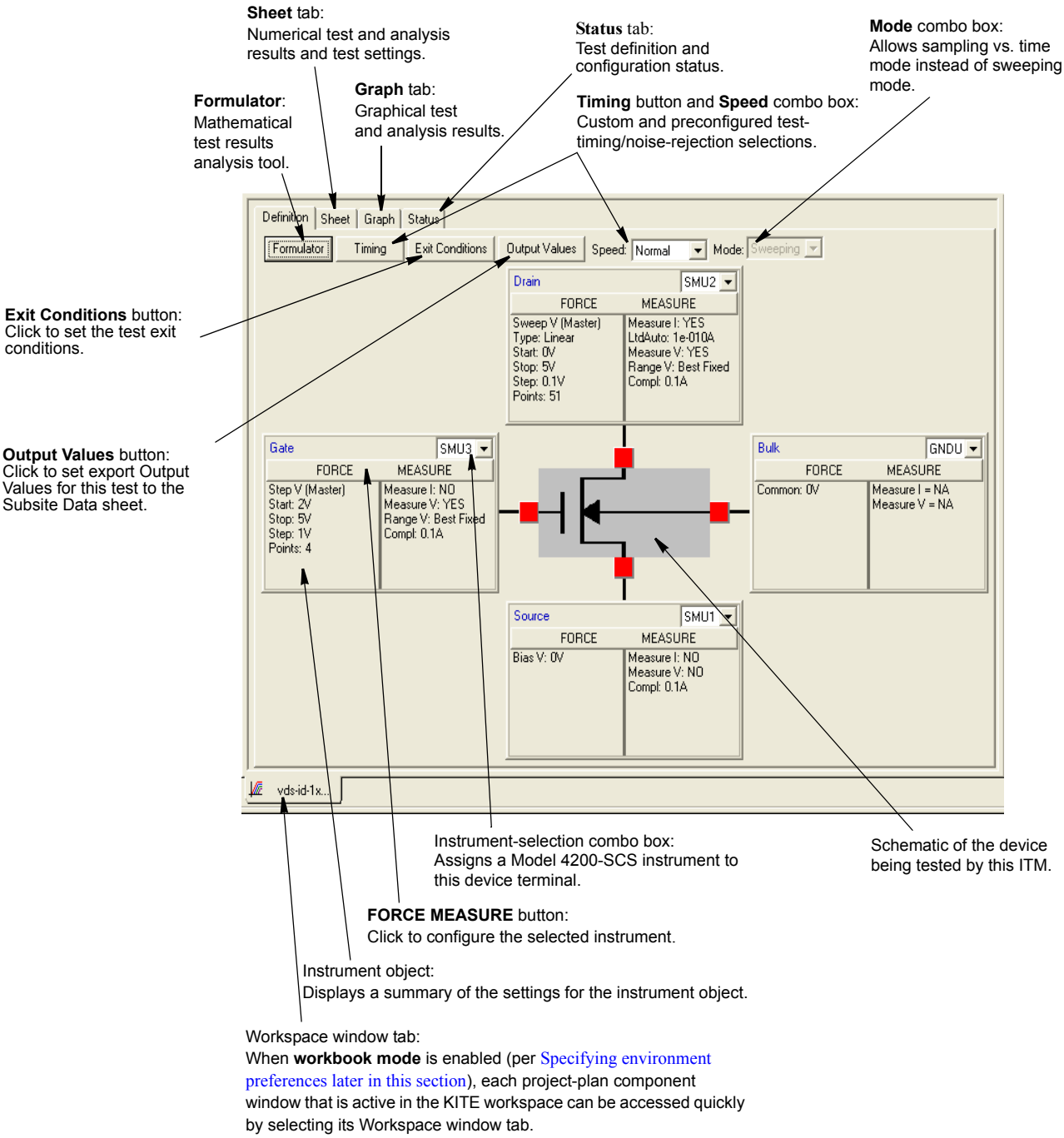
   Test sequence with real time plotting:

```
initialize instrument;
setup single point measurement mode;
setup single trigger mode;
turn on output; calculate number
of data point;
// measurement loop
for (index = 0; index < numpoint; index ++)
{
setup force value in each step;
take measurement;
// transfer data for real time plotting
PostDataDouble("Variable Name", MeasureArray[index]);
}
post measurement clean up;
done;
```

Defining an ITM

An ITM is defined by the ITM **Definition** tab (displayed by double-clicking on the ITM name in the Project Navigator). Figure 6-4 illustrates and explains the ITM **Definition** tab (Figure 6-4 defines the **"vds-id"** ITM, one of the ITMs in the example Project Plan that was shown in Figure 6-2 and Figure 6-3). See Specifying environment preferences later in this section.

Figure 6-4
ITM Definition tab



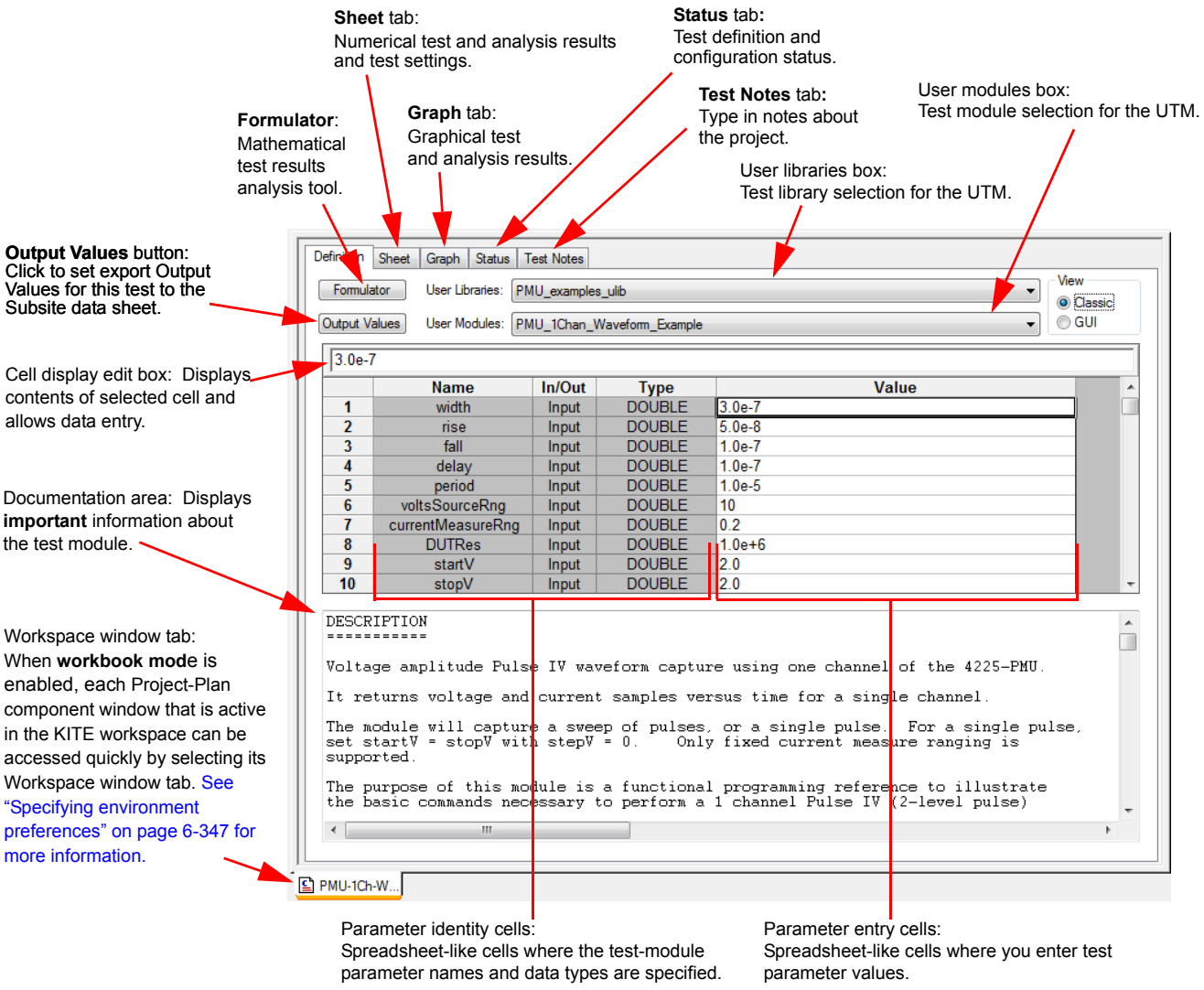
For details about defining and configuring an ITM, refer to Configuring the Project Plan ITMs later in this section.

Defining a UTM

A UTM is defined using the UTM definition tab (displayed by double-clicking the UTM name in the project navigator). Figure 6-5 and Figure 6-6 illustrate and explain the two versions of the UTM definition tab. Figure 6-5 defines the `PMU-1Ch-Wfm` UTM in the `PMU-DUT-Examples` project. The `PMU-DUT-Examples` project has several UTMs for use with the Model 4225-PMU Dual Channel Pulse I-V instrument card.

Figure 6-5 illustrates the classic view (table-based) version of the of the UTM definition; Figure 6-6 shows the GUI view. The GUI view does not change the operation of the UTM or the overall project execution. The GUI view (Figure 6-6) utilizes a variety of ways to enter values: typing text in an edit box, clicking on a drop-down list, or selecting a check box or option button. The UTM GUI view simplifies the presentation of the UTM test parameters by not displaying some of the less-used parameters. Entering a value in the GUI view also places the value into the table-based classic definition; entering a value in the classic view also places it in the GUI view. If there is a parameter in the table, but not in the GUI definition, then the table determines the value used by the UTM.

Figure 6-5
UTM classic definition tab



A UTM is created and configured by first selecting a user library and user module, and then entering parameter values. For details about defining and configuring a UTM, refer to the See “Configuring the UTMs” on page 6-147 for more information.

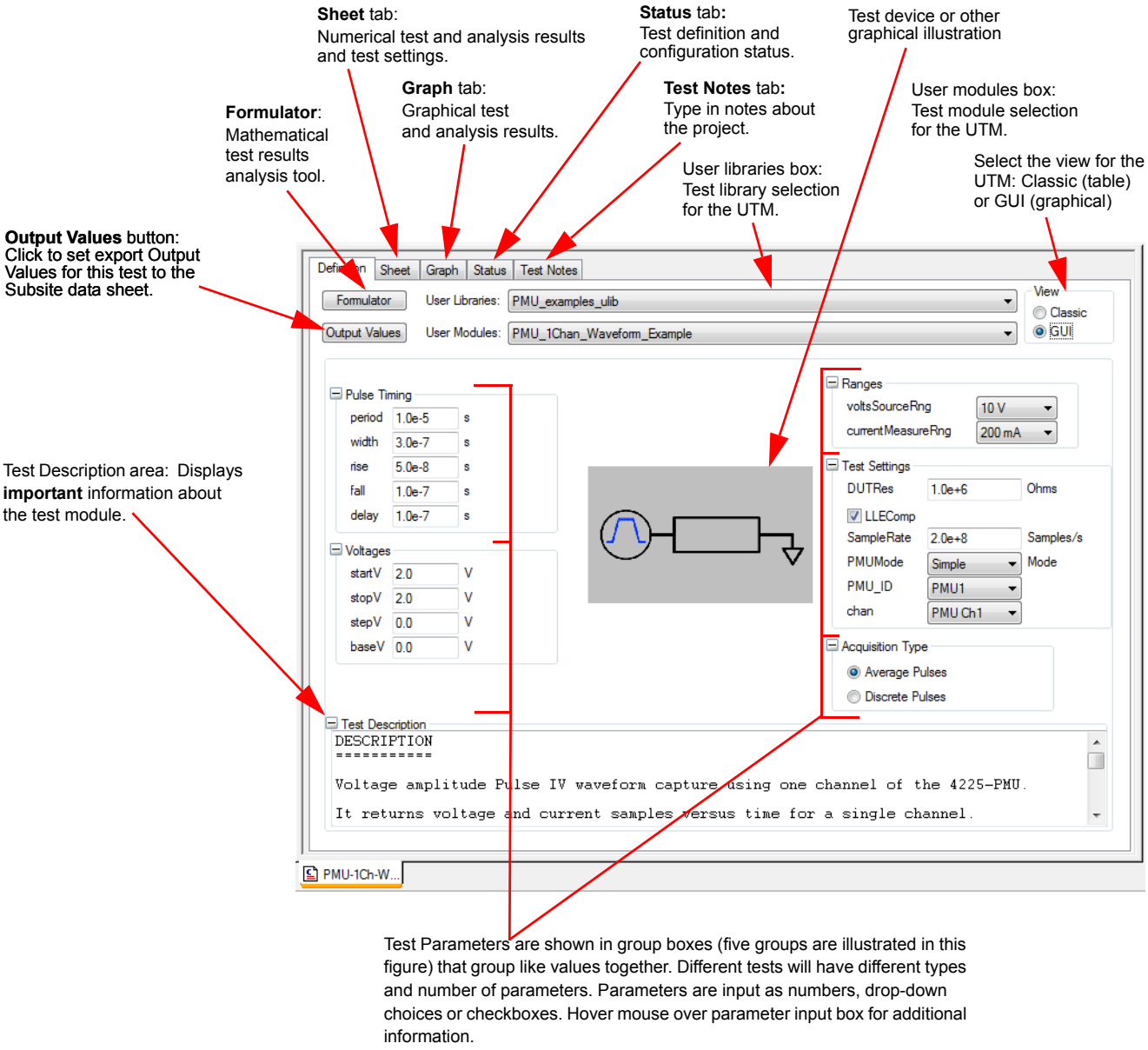
Using the UTM GUI view

To understand the test, read the explanation of the test in the Test Description area (see Figure 6-6 on page 6-21). Note that the UTM GUI view will typically hide the less-used parameters; these parameters are available in the classic view. If a parameter's use is unclear, check if the parameter has any hovertext help (position the mouse pointer over the parameter entry field). Additional information about parameter usage may also be in the Test Notes tab, or in the test device or other graphical illustration area.

If a group box has a plus sign (+) in it, then the box is collapsed; click the + to expand the box. A box will only expand when there is sufficient space to show all of the parameters within the group. In cases where there are many parameters in the UTM GUI view, it may be necessary to collapse a group above or below to allow the box to expand. To collapse an expanded group box, click the (-).

For a user module, the exact controls and parameter grouping of a UTM GUI view are determined by the GUI view definition. This is typically created by the same person that created the underlying user module.

Figure 6-6
UTM GUI definition tab

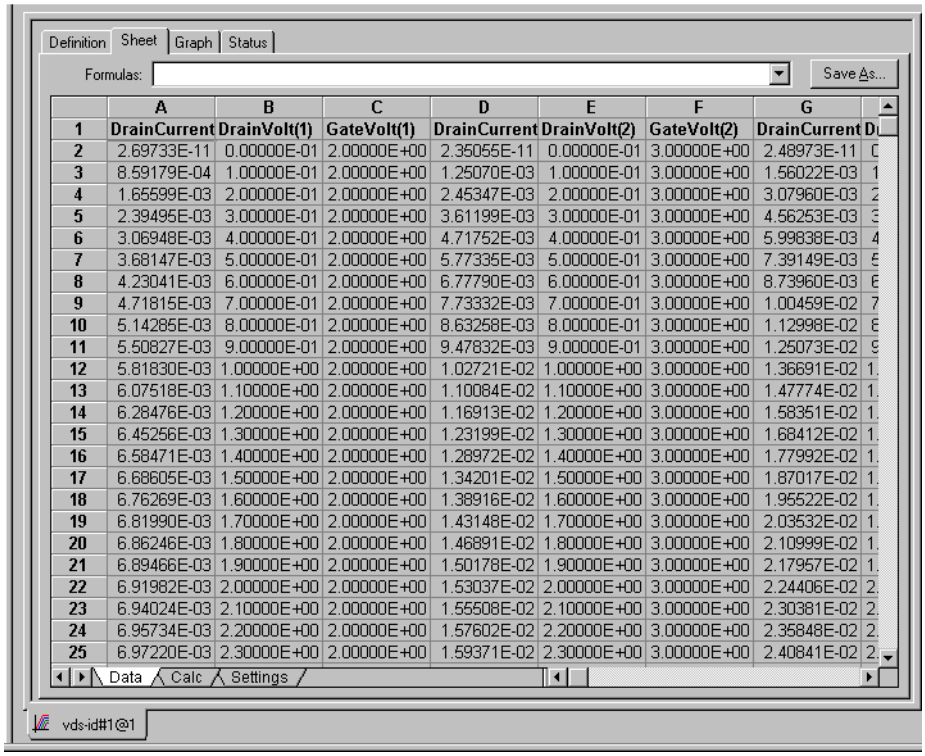


A UTM is created and configured by selecting a user library and user module and then entering parameter values. For details about defining and configuring a UTM, refer to Configuring the UTMs later in this section.

Viewing ITM or UTM results numerically: The Sheet tab Data worksheet

Two other tabs that are accessible from an ITM or UTM window display data and data-analysis results. One of these, the **Sheet** tab displays the data numerically on the Microsoft Excel-compatible **Data** worksheet. Figure 6-7 shows a **Sheet** tab **Data** worksheet containing data generated by the **"vds-id"** ITM, one of the ITMs in the example Project Plan that was shown in Figure 6-2 and Figure 6-3.

Figure 6-7
Sheet tab Data worksheet



	A	B	C	D	E	F	G
1	DrainCurrent	DrainVolt(1)	GateVolt(1)	DrainCurrent	DrainVolt(2)	GateVolt(2)	DrainCurrent
2	2.69733E-11	0.00000E-01	2.00000E+00	2.35055E-11	0.00000E-01	3.00000E+00	2.48973E-11
3	8.59179E-04	1.00000E-01	2.00000E+00	1.25070E-03	1.00000E-01	3.00000E+00	1.56022E-03
4	1.65599E-03	2.00000E-01	2.00000E+00	2.45347E-03	2.00000E-01	3.00000E+00	3.07960E-03
5	2.39495E-03	3.00000E-01	2.00000E+00	3.61199E-03	3.00000E-01	3.00000E+00	4.56253E-03
6	3.06948E-03	4.00000E-01	2.00000E+00	4.71752E-03	4.00000E-01	3.00000E+00	5.99838E-03
7	3.68147E-03	5.00000E-01	2.00000E+00	5.77335E-03	5.00000E-01	3.00000E+00	7.39149E-03
8	4.23041E-03	6.00000E-01	2.00000E+00	6.77790E-03	6.00000E-01	3.00000E+00	8.73960E-03
9	4.71815E-03	7.00000E-01	2.00000E+00	7.73332E-03	7.00000E-01	3.00000E+00	1.00459E-02
10	5.14285E-03	8.00000E-01	2.00000E+00	8.63258E-03	8.00000E-01	3.00000E+00	1.12998E-02
11	5.50827E-03	9.00000E-01	2.00000E+00	9.47832E-03	9.00000E-01	3.00000E+00	1.25073E-02
12	5.81830E-03	1.00000E+00	2.00000E+00	1.02721E-02	1.00000E+00	3.00000E+00	1.36691E-02
13	6.07518E-03	1.10000E+00	2.00000E+00	1.10084E-02	1.10000E+00	3.00000E+00	1.47774E-02
14	6.28476E-03	1.20000E+00	2.00000E+00	1.16913E-02	1.20000E+00	3.00000E+00	1.58351E-02
15	6.45256E-03	1.30000E+00	2.00000E+00	1.23199E-02	1.30000E+00	3.00000E+00	1.68412E-02
16	6.58471E-03	1.40000E+00	2.00000E+00	1.28972E-02	1.40000E+00	3.00000E+00	1.77992E-02
17	6.68605E-03	1.50000E+00	2.00000E+00	1.34201E-02	1.50000E+00	3.00000E+00	1.87017E-02
18	6.76269E-03	1.60000E+00	2.00000E+00	1.38916E-02	1.60000E+00	3.00000E+00	1.95522E-02
19	6.81990E-03	1.70000E+00	2.00000E+00	1.43148E-02	1.70000E+00	3.00000E+00	2.03532E-02
20	6.86246E-03	1.80000E+00	2.00000E+00	1.46891E-02	1.80000E+00	3.00000E+00	2.10999E-02
21	6.89466E-03	1.90000E+00	2.00000E+00	1.50178E-02	1.90000E+00	3.00000E+00	2.17957E-02
22	6.91982E-03	2.00000E+00	2.00000E+00	1.53037E-02	2.00000E+00	3.00000E+00	2.24406E-02
23	6.94024E-03	2.10000E+00	2.00000E+00	1.55508E-02	2.10000E+00	3.00000E+00	2.30381E-02
24	6.95734E-03	2.20000E+00	2.00000E+00	1.57602E-02	2.20000E+00	3.00000E+00	2.35848E-02
25	6.97220E-03	2.30000E+00	2.00000E+00	1.59371E-02	2.30000E+00	3.00000E+00	2.40841E-02

Note that the Workspace window tab at the bottom of the **Sheet** tab **Data** worksheet identifies the ITM that generated the data.

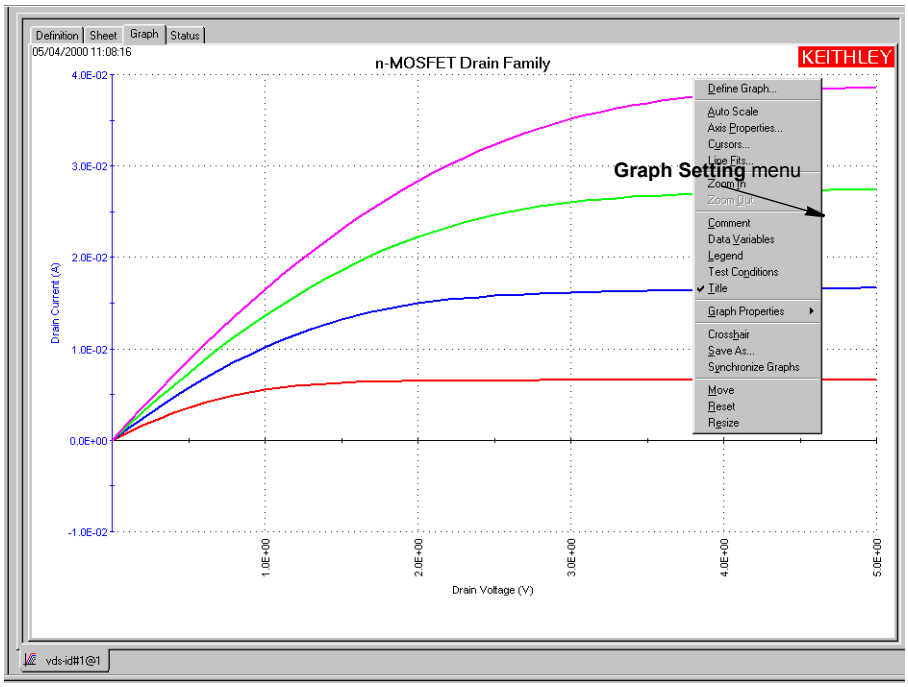
For more information about the **Sheet** tab, refer to Displaying and analyzing data using the Sheet tab later in this section.

Viewing ITM or UTM results graphically: The Graph tab

The **Graph** tab displays user-specified data graphically in a user-specified format. A pop-up menu (displayed when the **Graph** tab is displayed, by right-clicking on the **Graph** tab or selecting **Tools > Graph Settings**) accesses configuration parameters for the plot.

Figure 6-8 shows a **Graph** tab containing data generated by the **vds-id** ITM, one of the ITMs in the example Project Plan that was shown in Figure 6-2 and Figure 6-3.

Figure 6-8
Graph tab



Note that the Workspace window tab at the bottom of the **Graph** tab identifies the ITM (or UTM) that generated the data.

For more information about the **Graph** tab, refer to Viewing data using the Graph tab later in this section.

Developing and using user libraries for UTMs

Developing test modules

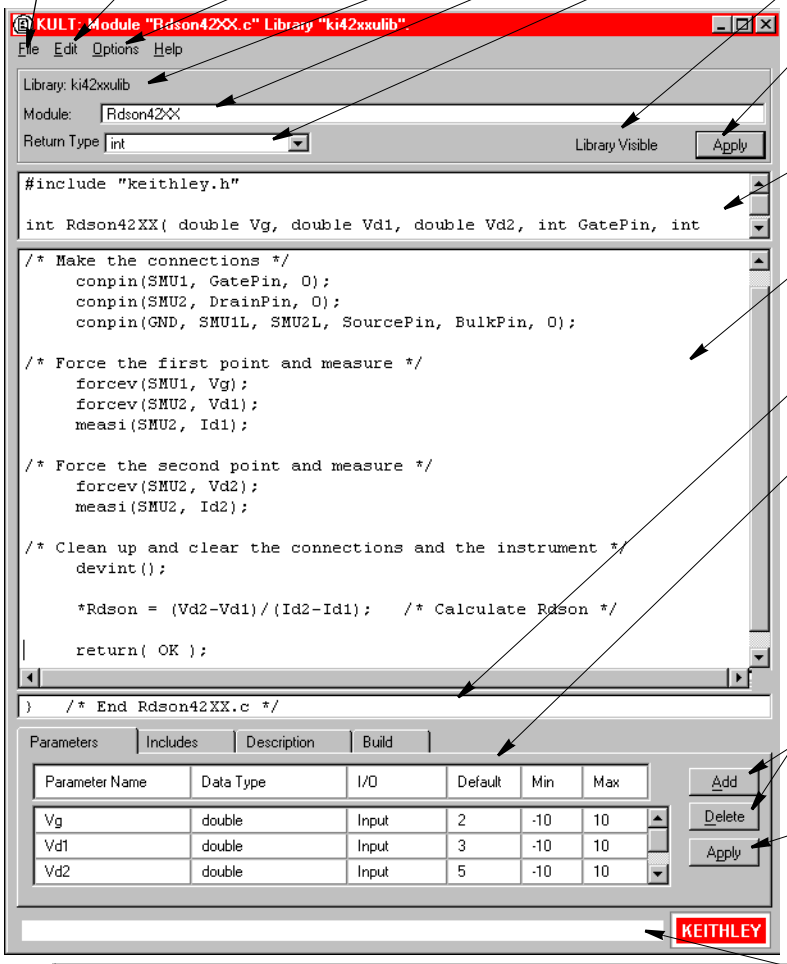
A UTM is created and configured by selecting a user library and user module and then entering parameter values. Keithley provides several user-module user libraries. You can create others to meet special needs.

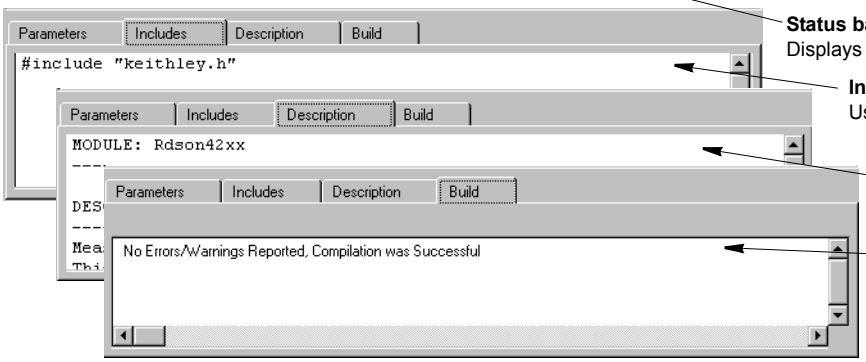
A user module is a C-language function (subroutine). A user library is a dynamic link library (DLL) of user modules that are compiled and linked using the Keithley User Library Tool (KULT). Several user libraries are provided with the Model 4200-SCS. You can use these as-is, customize them (possible in most cases), or create completely new ones. Most user modules contain functions from the Keithley-supplied Linear Parametric Test Library (LPTLib), as well as ANSI-C functions. All user modules are created and built using KULT.

Figure 6-9 illustrates the KULT programming and definition interface for a user module, in this case for the RDSon42XX user module, which is located in the Keithley Instruments-supplied ki42xxulib user library.

NOTE *RDSon42XX is the user module that is associated with the* ***res_drain-to-source*** *UTM, one of the tests in the example Project Plan that was shown in Figure 6-2 and Figure 6-3. It is also the user module that was specified per Figure 6-5. It measures the drain-to-source resistance of an FET at saturation.*

Figure 6-9
KULT interface overview





File menu:
Used to: open and close libraries and modules; save, copy, and delete modules; and so on..

Edit menu:
Used to: cut, copy and paste; select all; undo and redo.

Options menu:
Used to: compile the active module; add/ update the user module to the active user library; hide the module to make it unavailable to KITE user.

Library:
Displays the name of the active library

Module box:
Displays the name of the active module.

Return Type combo box:
Used to select the output data type from one of the following options: char, float, double, int, long, void.

Library Visible or **Library Hidden** display:
Indicates if library is available or unavailable to KITE. Visibility is controlled using the **Options** menu.

Apply button (one of two, both of which function identically):
Used to update active module to reflect additions and changes. Creates new active module when **Module** name is changed.

Module-parameter display area:
Displays—only—the includes, defines and function prototype for the module, to reflect entries in the **Parameters** entry tab and **Includes** entry tab areas.

Module code entry area:
Displays the C-code of the active module and, with its integral text editor, enables code development and editing.

Terminating brace area:
Shows terminating brace for the module code; automatically entered when you click either **Apply** button.

Parameters entry tab area:
Used to do the following for each module I/O parameter:

- Enter the parameter name.
- Select the parameter data type from a list of data types, data-pointer types, and array types [outputs must always be pointers (char *, float *, double *, and so on, or arrays).
- Select whether the parameter is an input or output (only selectable for pointers and arrays; others always inputs). Optionally enter default, max, and min parameter values.

Add and **Delete** buttons:
Used to add or delete a selected parameter from **Parameters** entry tab area.

Apply button (one of two, both of which function identically):
Used to update active module to reflect additions and changes. Creates new active module when **Module** name is changed.

Status bar:
Displays description of window area at cursor location.

Includes entry tab area:
Used to enter the module's include and define statements.

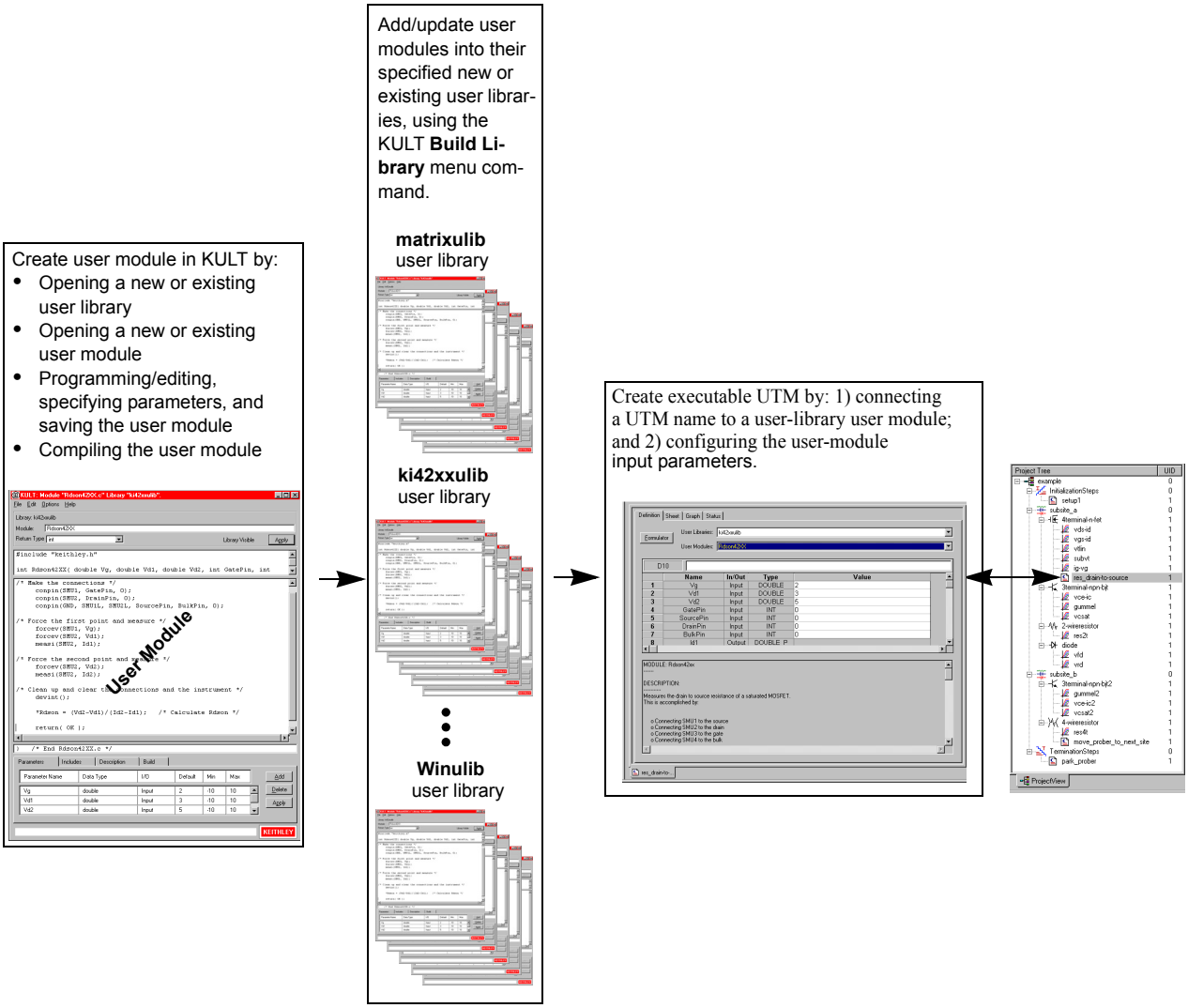
Descriptions entry tab area:
Used to document the active module.

Build tab area:
Displays status and error messages for the Compile and Library Build operations. A double-click on an error message highlights the trouble spot in the module code area.

Creating and using user libraries

Figure 6-10 summarizes and ties together the creation of user libraries, user modules, and UTMs.

Figure 6-10
Creating and using user libraries



Basic test execution

NOTE *If KITE detects an above-normal temperature condition at any SMU, it protects system outputs by preventing or aborting a run and reporting the condition in the message area of KITE window. If the condition occurs when a run is attempted, KITE prohibits execution. If the condition occurs during a run, KITE aborts the run(Not in Ref Man).*

Project Navigator Checkboxes

As shown in Figure 6-11, each component of the Project Plan has a checkbox. A check mark in a box indicates that the test or plan is enabled. The absence of a check mark indicates that the test or plan is disabled. Clicking a checkbox either inserts a check mark to enable or removes a check mark to disable. Only enabled (check marked) tests or plans can be run.

There is interaction between the Project Navigator Checkboxes and is explained by the following actions:

Tests (ITMs and UTMs)

- A check mark can be inserted or removed for any test.
- Inserting a check mark for a test also inserts a check mark for its Device Plan, its Subsite Plan, and the Project Plan.

Device Plan

- Removing a check mark for a Device Plan also removes the check marks for all its tests.
- Inserting a check mark for a Device Plan also inserts check marks for all its tests.
- Removing the check marks for all the tests in the Device Plan, also removes the check mark for the Device Plan.

Subsite Plan

- Removing a check mark for a Subsite Plan also removes the check marks for all its Device Plans and tests.
- Inserting a check mark for a Subsite Plan also inserts check marks for all its Device Plans and tests.
- Removing the check marks for all the tests in the Subsite Plan, also removes the check mark for the Subsite Plan.

Initialization and Termination Steps

- Removing a check mark for Initialization or Termination Steps also removes the check marks for all its UTMs.
- Inserting a check mark for Initialization or Termination Steps also inserts check marks for all its UTMs.
- Removing the check marks for all the UTMs in the Initialization or Termination Steps, also removes the check mark for the Initialization or Termination Steps.

Project Plan

- Removing a check mark for a Project Plan also removes the check marks for all its plans and tests.
- Inserting a check mark for a Project Plan also inserts check marks for all its plans and tests.

- Removing the check marks for all the tests in the project, also removes the check mark for the Project Plan.

NOTE *Figure 6-17 shows an example of Project Plan structure that shows a mix of enabled and disabled tests.*

Executing an individual test

An enabled test must be selected before it can be run.

Selecting a test

An enabled (check marked) ITM or UTM is selected by clicking the test in the Project Navigator (see Figure 6-11). The **Run Test/Plan** button (▶) turns green to indicate that the test is enabled and ready to be run. Also, the selected-test name is displayed in the Test/Plan Indicator box located above the Project Navigator.

The test can also be selected by double-clicking the test in the Project Navigator. The double-click action places the appropriate ITM or UTM window in the KITE workspace. The ITM and UTM **Definition** tabs show the test configurations.[1] See Figure 6-4 and Figure 6-5.

NOTE *Before executing a test for which data must be labeled with a specific site number, refer to Assigning a site-number label to individual test and test-sequence data later in this section.*

Running the test

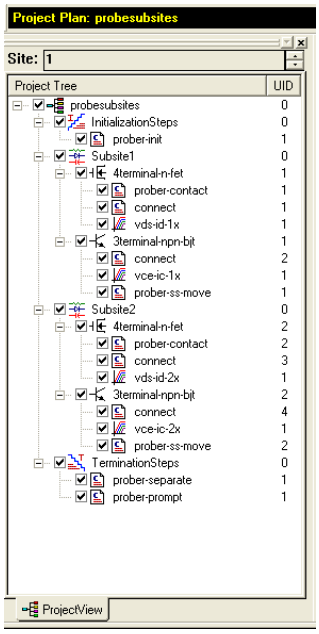
The selected test is run by clicking the green **Run Test/Plan** toolbar button (▶), selecting **Run** in the **Run** menu, or pressing the **F6** keyboard key. While the test is running, test data is placed in a data sheet. ITM data may usually be viewed dynamically, both numerically and, if the **Graph** tab has been appropriately configured, graphically (new UTM data may be viewed only at the end of a test). In the Message area of the KITE window, time stamps indicate start time, stop time, and total execution time.

NOTE *You can also start a test by pressing the F6 keyboard key. You can abort a test by clicking the red **Abort Test/Plan** toolbar button, by selecting **Abort** in the **Run** menu, or by pressing the **PAUSE/BREAK** keyboard key.*

For detailed information on running individual tests, see Run execution of individual tests and test sequences later in this section.

1. For details on using the ITM and UTM **Definition** tabs, see Configuring the Project Plan ITMs, and Configuring the UTMs later in this section.

Figure 6-11
Example Project Plan



Executing an individual test sequence

A test sequence can include all of the tests in a Device Plan or a Subsite Plan. For example, if you run the 3terminal-npn-bjt Device Plan in Figure 6-11,

then the ITMs for that plan execute in the following sequence:

vce-ic > gummel > vcsat

When a Subsite Plan is run, all the tests in that plan execute in the order presented in the Project Navigator. For example, if you run the **subsite_b** plan in Figure 6-11, then the tests for that plan execute in the following sequence:

gummel2 > vce-ic2 > vcsat2 > res4t > move_prober_to_next_site

---

NOTE *Before executing a test sequence for which data must be labeled with a specific site number, refer to Assigning a site-number label to individual test and test-sequence data later in this section.*

A test sequence is run by: 1) enabling (with check marks) the plan or tests that you wish run; 2) selecting the Subsite Plan or Device Plan in the Project Navigator; and 3) clicking the green **Run Test/Plan** toolbar button (▶), selecting **Run** in the **Run** menu, or pressing the **F6** keyboard key.

For detailed information on running individual test sequences, see Run execution of individual tests and test sequences later in this section.

Executing appended tests and test sequences

With **Run** execution, described above, there is only one **Data** worksheet for each specific test. This **Data** worksheet contains the data from the last run of the test; each new run updates the worksheet. KITE also provides **Append** execution, which generates multiple worksheets for multiple runs of a specific test, in addition to the **Data** sheet generated by **Run** execution. Also, in the **Graph** tab, the **Append** data curves for a test append to (layer on top of) the **Run** data curves.

The **Append** mode may be applied to an entire test sequence (a Device Plan or Subsite Plan) as well as to a solitary test. Each time the sequence is run, the results of each test in the sequence appends to the appropriate graph.

Any of the following actions result in an **Append** execution of a selected test: clicking the green-in-yellow **Append Data** toolbar button ( ), selecting **Append** in the **Run** menu, or simultaneously pressing the **SHIFT** + **F6** keyboard keys.

For details about the **Append** mode, refer to Append execution of tests, test sequences, and Project Plans, and Appending curves from multiple runs on a single graph later in this section.

Assigning a site-number label to individual test and test-sequence data

If you run an individual test or test sequence at a specific site that must be identified in the data, then the KITE site-number data label must match the prober site number. This applies particularly when executing individual tests and sequences at multiple sites on a wafer.[2] However, it also applies whenever you want to generate, identify, and retain distinct data files for the test(s).

Just before executing the individual test or test sequence, specify the site-number label for the collected data by way of the Site Navigator, using its spin buttons (small arrows). The Site Navigator is located at the top of the Project Navigator, as shown in Figure 6-15.

For example, before running a test or test sequence at site 3, set the Site Navigator to **3** and move the prober to site 3. Then, when you run the test or test sequence, each data file that is generated at site 3 will be labeled with the correct site number (for site labeling conventions, refer to Test data file naming conventions later in this section).

NOTE *The Site Navigator does not automatically specify which site a prober moves to. It only allows you to specify the site-label for the data that is collected when you run a single instance of a test or sequence of tests.*

For details on executing individual tests and test-sequences, refer to Run execution of individual tests and test sequences later in this section.

Executing an entire Project Plan

When a Project Plan is run, all the enabled plans and tests in the Project Plan execute. For example, if you run the example Project Plan in Figure 6-11, all Project Plan components execute in order.

Components of a Project Plan are enabled by inserting check marks in the checkboxes for the plans and tests to be run. A Project Plan is selected to run by clicking the enabled plan in the Project Navigator (plan name appears above it in the Test/Plan Indicator box). An entire Project Plan is run by: 1) specifying the site label for the data (refer to Multi-site setup) and then; 2) clicking the green **Run Test/Plan** toolbar button, selecting **Run** in the **Run** menu, or pressing the F6 keyboard key.

As each individual test runs, the test name appears in the Test/Plan Indicator box. With an ITM window open in the KITE workspace, you can view the ITM data being placed in its data sheet or being graphed.

For details on running entire Project Plans, refer to 'Run' execution of Project Plans later in this section.

2. To specify site labels when executing entire Project Plans, refer to Multi-site Project Plan execution.

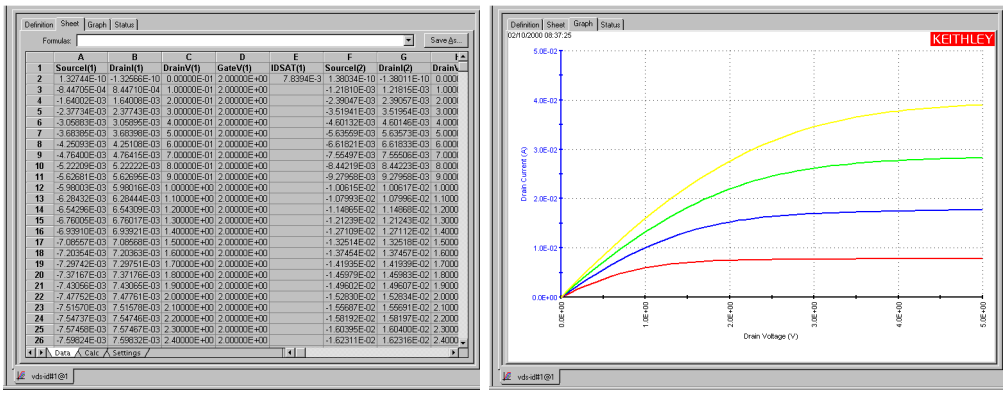
Test data

Data for an ITM or UTM is placed into a Microsoft Excel-compatible data sheet and, after axes are defined, may be graphed. The **Sheet** tab **Data** worksheet and the **Graph** tab graphing tools are accessed in the KITE workspace by: 1) double-clicking the test name in the Project Navigator to open the ITM or UTM window; and then 2) clicking the **Sheet** or **Graph** tab.

Data analysis and graphing tools

Graph and Sheet tabs: Clicking the **Sheet** tab displays the **Data** worksheet; clicking the **Graph** tab displays the graph. Figure 6-12 shows typical **Sheet** and **Graph** tabs for a test (in this case for the example Project Plan **"vds-id"** test). While an ITM is running, you can watch the data populate the data sheet or graph.

Figure 6-12
Sheet tab Data worksheet and Graph tab

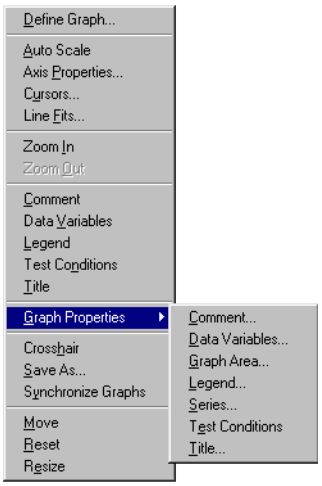


For detailed information on using the **Sheet** and **Graph** tabs, refer to Displaying and analyzing data using the Sheet tab and Viewing data using the Graph tab earlier in this section.

Graph tools: An assortment of graph tools is available to modify the graph characteristics, such as scale, axis properties, zoom view, legends, title, and comments. You can also save the graph as a bitmap (.bmp) file. With the graph displayed in the KITE workspace, open the graph menu, Figure 6-13, by right-clicking on an open area of the graph.

For detailed information on using the graph tools, see Viewing data using the Graph tab later in this section.

Figure 6-13
Graph settings menu



Formulator: Test data can be manipulated by way of user-defined formulas. When a formula is defined, the results are automatically added to the **Data** worksheet of the **Sheet** tab and may be selectively added to the **Graph** tab. To open the **Formulator**, click on the **Formulator** button in the **Definition** tab for the selected test.

For detailed information on the **Formulator**, refer to Analyzing test data using the Formulator later in this section.

Test data files

The data in the data sheet for each test is stored as a Microsoft Excel-compatible (.xls) file. This file is named and stored as described in the next two subsections (refer also to Test data file naming conventions later in this section).

Data file name

A file name is composed from the following:

- A test name: the ITM or UTM name.
- A Unique IDentifier number (**UID**). When a test is inserted into a Project Plan the first time, it is assigned **UID** #1. For every additional instance of a same-named test in the Project Plan, the **UID** number is incremented. Therefore, the second occurrence of that test is assigned UID #2, the third is assigned **UID** #3, and so on.
- A site number (refer to Multi-site Project Plan execution later in this section).

For example, data from the **"vds-id"** test in the **example** Project Plan is named as follows:

- `vds-id#1@1.xls`

where:

- `vds-id` is the test name.
- `#1` is the UID number.
- `@1` denotes the site number at which the test was executed.
- `.xls` is the extension for an Excel file.

NOTE *See Figure 6-16 for a graphical representation of data file names.*

Data file location

By default, test data files are stored on the Model 4200-SCS hard disk in the following directory:[3]

- `C:\S4200\kiuser\Projects\<ProjectName>\tests\data`

For example, the following path accesses the vds-id test data file for the **example** project:

- `C:\S4200\kiuser\Projects\example\tests\data\vds-id#1@1.xls`

For more information on test data file structure, refer to the Tests subdirectory in the User's Manual.

Multi-site Project Plan execution

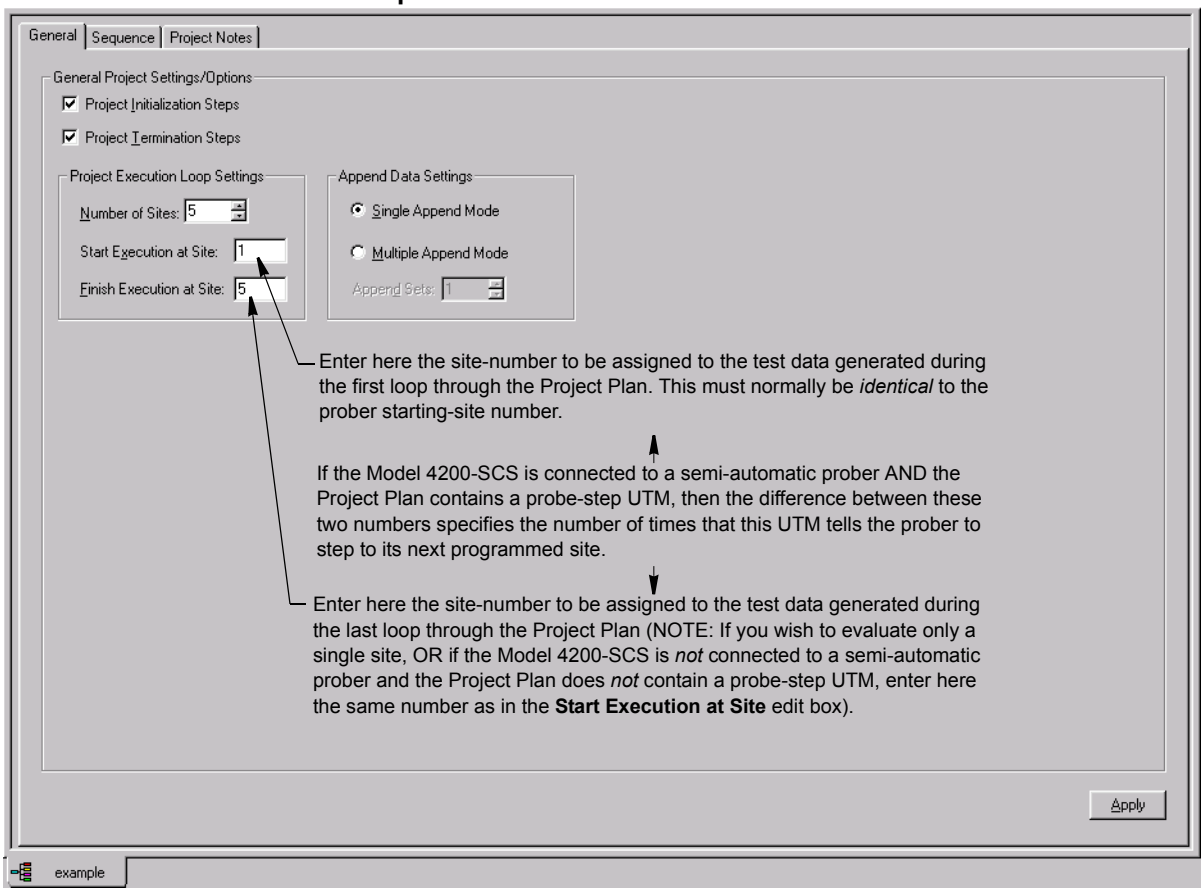
For multi-site Project Plan execution, all tests in the entire Project Plan are repeated for a series of wafer sites.

Multi-site setup

Multi-site execution is configured from the KITE Project window. This Project window is placed in the KITE workspace by double-clicking the project name in the Project Navigator. Figure 6-14 shows the KITE Project window for the example Project Plan.

3. You can also set up alternate user directories, including personal directories such as C:\S4200\YourName that include C:\S4200\YourName\Projects\<ProjectName>\.

Figure 6-14
Multi-site execution setup



CAUTION **In the Project window, do not change the settings on the Project Initialization Steps or Project Termination Steps checkboxes. They are to be used only when building or modifying a Project Plan. Typically, UTMs under InitializationSteps and TerminationSteps in the Project Plan perform such tasks as external instrument initialization and prober setup and parking. Unchecking Project Initialization Steps or Project Termination Steps checkboxes, and then clicking Apply, deletes the UTMs. Checking these checkboxes does not add UTMs.**

Specify the **Start Execution at Site** and the **Finish Execution at Site** numbers as explained in the callouts of Figure 6-14. **Finish Execution at Site** must be less than or equal to **Number of Sites,** which specifies the *maximum* number of sites that can be tested. Apply these specifications to the Project Plan by clicking **Apply**.

NOTE *As described in Figure 6-14, you must specify, in the Project window, the site-number(s) with which collected data is to be labeled. Before execution, you must also independently position the prober such that the first site to be evaluated on the wafer is identical to the first site that is specified in the Project window. This requirement applies whether you use a manual or semi-automatic prober and whether you evaluate one site or several.*

If you use a semi-automatic prober, understand that a KITE probe-step UTM only

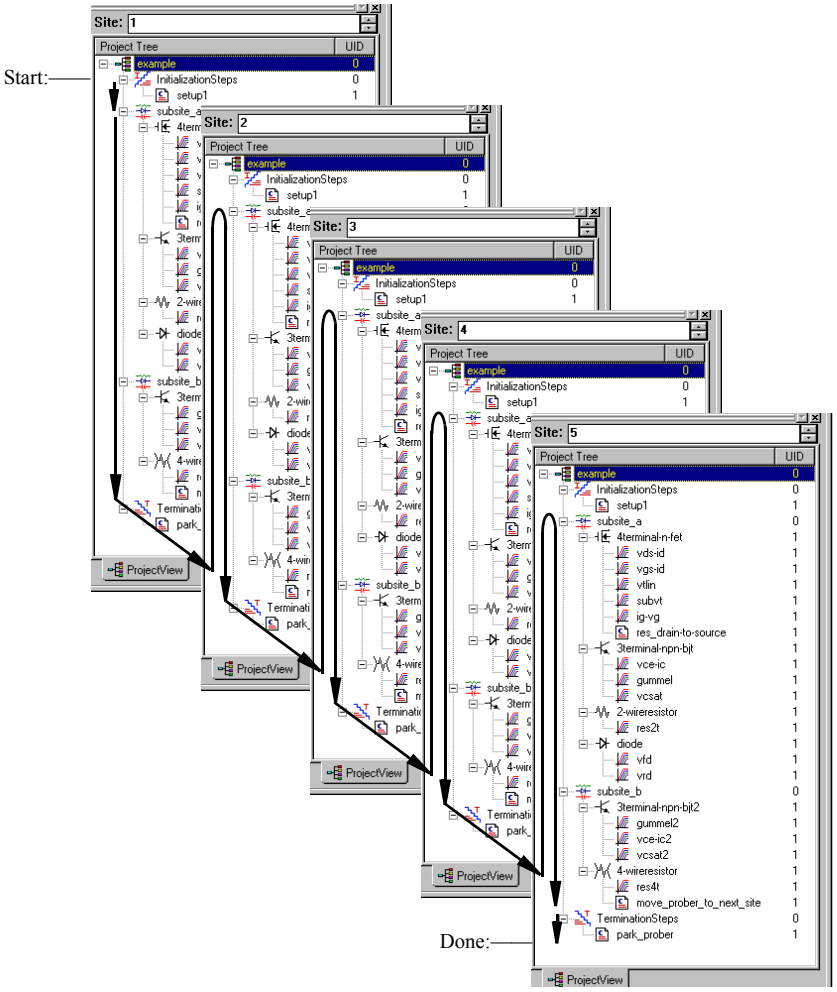
triggers movements that are already programmed in the prober controller. Each execution of the UTM advances the probe to the next site in this programmed sequence. Site numbers are not communicated between the prober and KITE. Therefore, if you evaluate multiple sites, the range of site numbers that you specify in the KITE Project window must agree with the sequence of site numbers in the prober-controller program.

Multi-site execution

A multi-site test sequence is selected to run by clicking the Project Plan name in the Project Navigator (Project Plan name appears in the Test/Plan Indicator box). The test sequence is started by clicking the green **Run Test/Plan** toolbar button or selecting **Run** in the **Run** menu.

Figure 6-15 shows the test sequence for the example Project Plan, which is configured to test five sites. At the start, the **InitializationSteps** are performed, and then the two Subsite Plans are executed for site 1. Then the prober moves to site 2, and the two Subsite Plans are executed again. This **move prober** –> **run Subsite Plans** process repeats for sites 3, 4, and 5. After the second Subsite Plan is finished at site 5, the **TerminationSteps** park the prober.

Figure 6-15
Multi-site test sequence



Multi-site test data

A set of data is generated for each of the selected sites. For example, five sets of data (one for each site) are generated for the example Project Plan test sequence shown in Figure 6-15. This subsection explains the following:

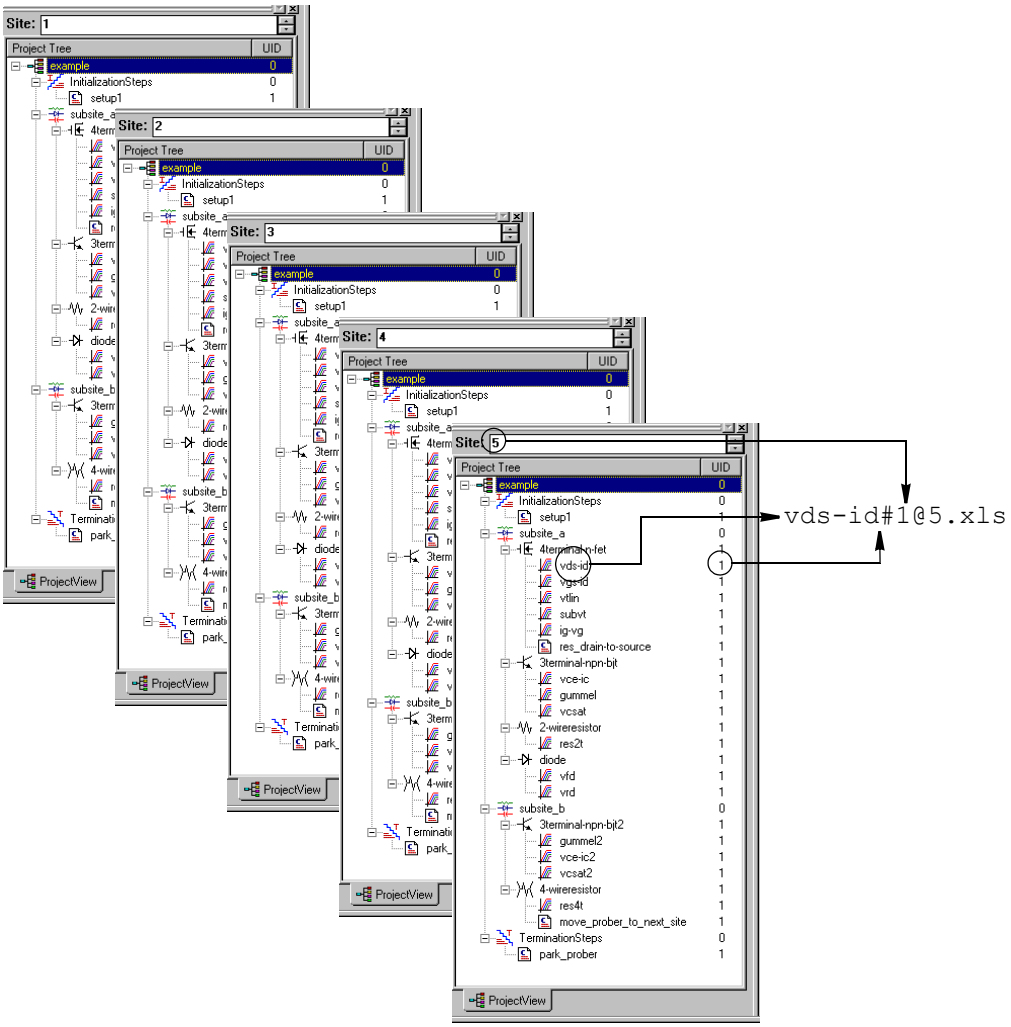
- Naming of the Excel-compatible data files (refer also to Test data files).
- Using the Site Navigator and Project Navigator to view the data collected at any site (the Site Navigator is located just above the Project Navigator, as shown in Figure 6-15).

Test data file naming conventions

If the Workbook environment mode is enabled (refer to Specifying environment preferences later in this section), the Workspace window tab at the bottom of an ITM or UTM window indicates: 1) the name of the ITM or UTM; 2) its **UID** (Unique IDentifier) number; and 3) the site number where the data was collected. For example, in the Workspace window tab at the bottom of the **Sheet** tab and **Graph** tab is labeled **vds-id#1@1**. The **"vds-id"** is the ITM name, the **#1** indicates a level-1 **UID** (Unique Identifier) number, and the **@1** indicates that the data was collected at site 1.

Figure 6-16 illustrates naming of a specific test data file (in this case, for an ITM, but equally applicable for a UTM).

Figure 6-16
Workspace-window tab name and data file name format



All data that is generated by a test is stored in a file having the same naming convention as for an ITM/UTM Workspace window tab, except for the addition of an .xls (Microsoft Excel) file ex-tension. See Figure 6-16. Again, the name is made up of the test name, the UID number and the site number. The file name `vds-id#1@5.xls` indicates an Excel-compatible data file for the **vds-id** ITM having **UID** number **1** that was generated at site number **5**. If the Project Plan contained second and third instances of the **"vds-id"** ITM, their respective UID numbers would be 2 and 3, and their site 5 data files would be labeled `vds-id#2@5.xls` and `vds-id#3@5.xls`. Figure 6-16 also shows the source of the information on the Workspace window tab.

Test-data viewing, using the Site Navigator to specify site numbers

Use the Site Navigator to locate data to be viewed, specifying the site at which the data was collected. For example, to view data for an ITM that was executed at site 5, first set the Site Navigator to **5**, and then double-click the ITM in the Project Navigator. When the corresponding ITM window opens, then display the **Sheet** tab or **Graph** tab by clicking it. You need not wait for a test sequence or Project Plan execution to finish before navigating test data by way of the above procedure. You can select and view data sheets and graphs for any ITM at any site in real time, as the data is being acquired. The test name for the executing test, including UID number and site number, displays in the Test/Plan Indicator box.

For details on viewing test data, refer to Displaying and analyzing test results later in this section.

Understanding KITE

The Keithley Interactive Test Environment (KITE) is the main software component of the KTE Interactive software tool set. KITE is the primary user interface for the Keithley Model 4200-SCS Semiconductor Characterization System. KITE is a versatile tool that facilitates interactive characterization of an individual parametric test device or automated testing of an entire semiconductor wafer.

Two additional KTE Interactive software tools augment the capabilities of KITE, as follows:

- The Keithley User Library Tool (KULT) is used to create test modules, using the C programming language. These test modules can then be executed by KITE.
- The Keithley CONfiguration utility (KCON) is used to manage the configuration and interconnections between all of the test system components that are controlled by KITE.

A fourth KTE Interactive software tool, the Keithley External Control Interface (KXCI) allows the Model 4200-SCS to be controlled remotely by an external GPIB controller.

NOTE KXCI and KITE are mutually exclusive software tools. That is, KXCI and KITE cannot run simultaneously.

Beginning with KTE Interactive 6.0, two optional KTE Interactive tools have been added:

- The Keithley Pulse tool (KPulse) is a virtual front panel software application used to control the optional pulse generator cards. The dual-channel pulse generator cards are integrated inside the Model 4200-SCS mainframe.
- The Keithley Scope tool (KScope) is a virtual front panel software application used to control the optional scope card. The scope card is a dual-channel Digital Storage Oscilloscope that is integrated inside the Model 4200-SCS mainframe.

NOTE *Although KScope and KPulse can be launched at the same time as KITE, KScope/ KPulse and KITE cannot communicate with hardware simultaneously.*

Project defined

Users interact with KITE in the context of project. A project specifies the start-to-finish, repetitive and nonrepetitive actions and test locations involved in evaluating a semiconductor wafer (or other collection of circuits). Projects are both created and executed using the KITE graphical user interface.

NOTE *Refer also to Project Plan later in this section. The term "project" is sometimes used to refer to a project plan.*

Project components

Because KITE is most valuable for automatic characterization of semiconductor wafers, KITE projects are organized in a manner consistent with the organization of a modern semiconductor wafer. A project visits and evaluates locations on the wafer in the following logical hierarchy:

- Project
  - Sites
    - Subsites

- Devices
  - Tests

These are the primary components that make up a project. Two other components, initialization steps and termination steps, are discussed under Project structure later in this section. These components are defined contextually in the next subsections.

Sites

At the macroscopic level, one or more semiconductor dies are built up at a given wafer location. This location is comprised not only of end product dies, but usually has one or more parametric test structures or *subsites*. KITE refers to such a repeating pattern of dies and test structures as a *site*.

Subsites

The terminals of each device on a test structure are connected to a uniformly spaced series of contact pads. These pads are used to connect the devices to the probes of a prober. Every wafer location that the prober moves to and contacts at any one time is called a *subsite*, sometimes referred to as a Test Element Group or TEG.

The Model 4200-SCS hardware/KITE software combination was optimized to evaluate test structures, though it can be effectively used to evaluate dies and discrete components. KITE refers to each such test structure (or combination of test devices that are tested as a group) as a *subsite*.

Devices

As described in context under Sites, each test structure contains a series of devices to be characterized: transistors, diodes, resistors, capacitors, and so on. A switch matrix is used to connect the Model 4200-SCS sequentially if the SMUs cannot be connected to all devices simultaneously.

A *device* is also referred to as a test element, because subsites are often referred to as test structures or Test Element Groups (TEGs), which are composed of devices.

Tests

Once the test device is in position, KITE automatically conducts one or more specified tests for each device on the test structure. Each test generates data and, if desired, parametric curves. A test includes the following for each terminal of a device:

- The desired voltage or current forcing functions (stimuli).
- The desired voltage or current measurements.
- The associated data analyses and parameter extractions.

The combination of forcing functions and measurements is referred to as the test definition.

There are two classes of tests or test modules in KITE: Interactive Test Modules (ITMs) and User Test Module (UTMs). Both ITMs and UTMs share common data analysis functions, such as a Microsoft Excel-compatible data sheet and a real-time graph tool. Key differences between ITMs and UTMs include the following:

- **Interactive Test Module (ITM):** An ITM allows the user to define a test interactively by way of a graphical user interface.
- **User Test Module (UTM)**: A UTM is defined through the programming of its connected KULT created user module, but allows the user to configure key test parameters using a graphical user interface.

Differences between ITMs and UTMs were discussed in more detail under Overview of KITE earlier in this section.

Project structure

The entire series of operations of a project is structured in a hierarchical order that is determined by the Project Plan. Similarly, at lower levels in the hierarchy, the series of operations that is performed at each site, subsite, and device is determined by a Site Plan, Subsite Plan, or Device Plan. Each plan level is discussed individually below.

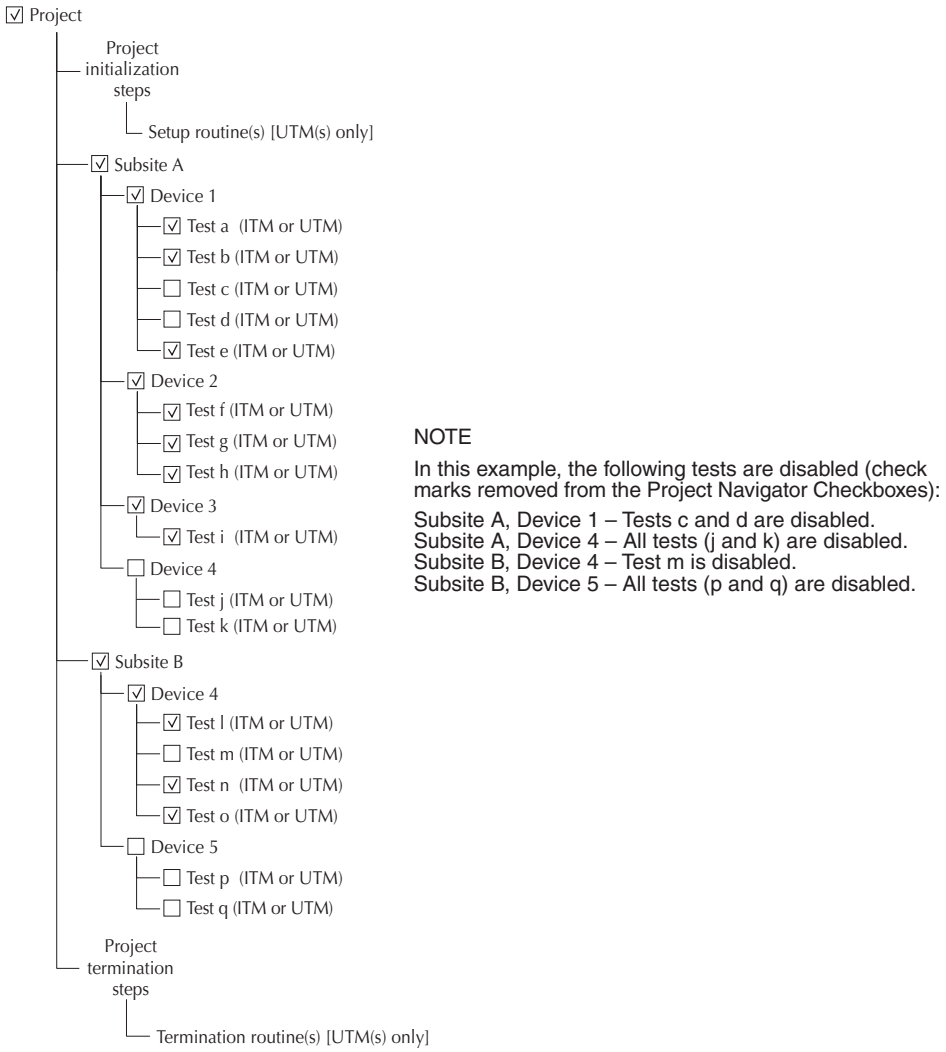
Project Plan

The following operations may be included in the Project Plan.

- Initialization of the project.
- Prober movement between project sites and subsites.
- If necessary, cycling electrical connections from the Model 4200-SCS between the devices of a subsite, using switch matrices.
- Execution of the tests for each device at each subsite.
- Termination of the project.

Figure 6-17 illustrates the hierarchy of an example Project Plan.

Figure 6-17
Project Plan hierarchy



The active Project Plan is displayed in the KITE Project Navigator, which is the window displayed at the left of the KITE main screen. See Figure 6-18. The Project Navigator provides the following:

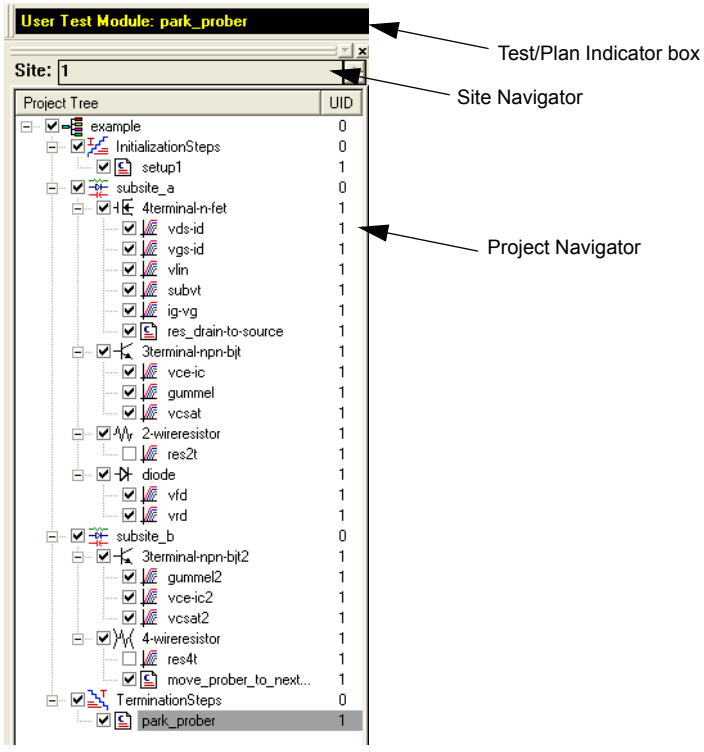
- Displays the Project Plan structure/hierarchy.
- Allows point-and-click opening of configuration screens and menus for each of the individual Project Plan components.
- Allows point-and-click selection of where Subsite Plans, Device Plans, and tests are to be added to or deleted from the Project Plan.
- Allows point-and-click control of the Project Navigator Checkboxes. Plans and tests are enabled to run by inserting check marks in the checkboxes. Removing check marks disables plans and tests (see NOTE in Figure 6-17).
- Allows point-and-click selection of individual Subsite Plans, Device Plans, or tests, exclusive of other parts of the Project Plan.

The combo box labeled *Site* directly above the Project Navigator is the Site Navigator. The Site Navigator does the following:

- Determines the site that newly acquired data will be assigned to when the **Run** button is clicked.
- Allows a site to be selected manually by using the spin buttons (the arrows at the right side of the Site Navigator edit box).

Figure 6-18 shows the Project Navigator and Site Navigator for a Project Plan called *example*. The **example** Project Plan has the same structure as illustrated in Figure 6-17. Note that the Project Plan name is displayed above the Site Navigator in the Test/Plan Indicator box.

Figure 6-18
Example Project Plan, as displayed in the Project Navigator



Lower-level plans within this hierarchy are discussed in Project structure later in this section.

Initialization and termination steps

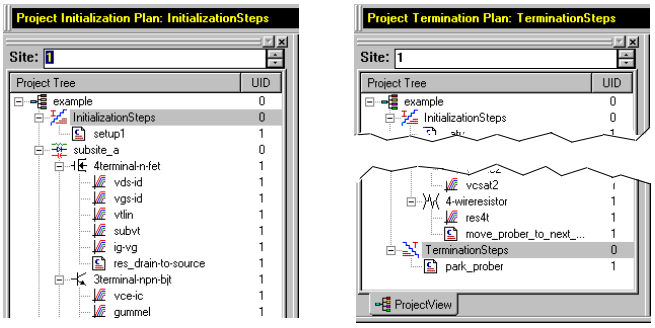
If the Project Plan is to be initialized and terminated automatically, initialization and termination steps are added. For example, initialization steps might be used to move a prober to a starting site and subsite or to initialize switch matrix connections to a starting configuration. Termination steps might be used to park a prober at a standby position, and so on.

Two distinct features distinguish initialization and termination steps from other elements of a Project Plan, as follows:

- An initialization or termination step is executed only at the Project Plan level, and only once at the beginning or end of the Project Plan. That is, initialization and termination steps are ignored as a Project Plan cycles between sites.
- Only UTMs are used for initialization and termination steps.

Figure 6-19 highlights how initialization and termination steps are displayed in the Project Navigator. Note that the selected initialization or termination step is also displayed above the Site Navigator in the Test/Plan Indicator box.

Figure 6-19
Display of initialization and termination steps in the Project Navigator

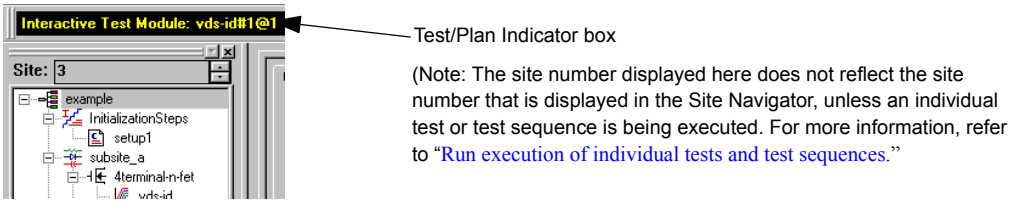


Site Plan

A Site Plan includes all of the Subsite Plans, Device Plans, and tests of the project. All sites are assumed to be identical. That is: 1) each site has the same type and number of subsites; and 2) sites are repeated across the wafer. The number of sites to be visited, typically by a prober, is specified in the Project window. The locations of sites to be visited are typically defined by the prober's software. However, the commands that initiate prober movement are defined by one or more prober-movement UTMs (as highlighted later in Figure 6-28).

No sites are displayed on the Project Navigator tree. Instead, the Test/Plan Indicator box displays the site number of the site currently being visited (as well as the test name and UID number) using the same coding as described under Test data file naming conventions earlier in this section. See Figure 6-20.

Figure 6-20
Active test and site number display in the Test/Plan Indicator box

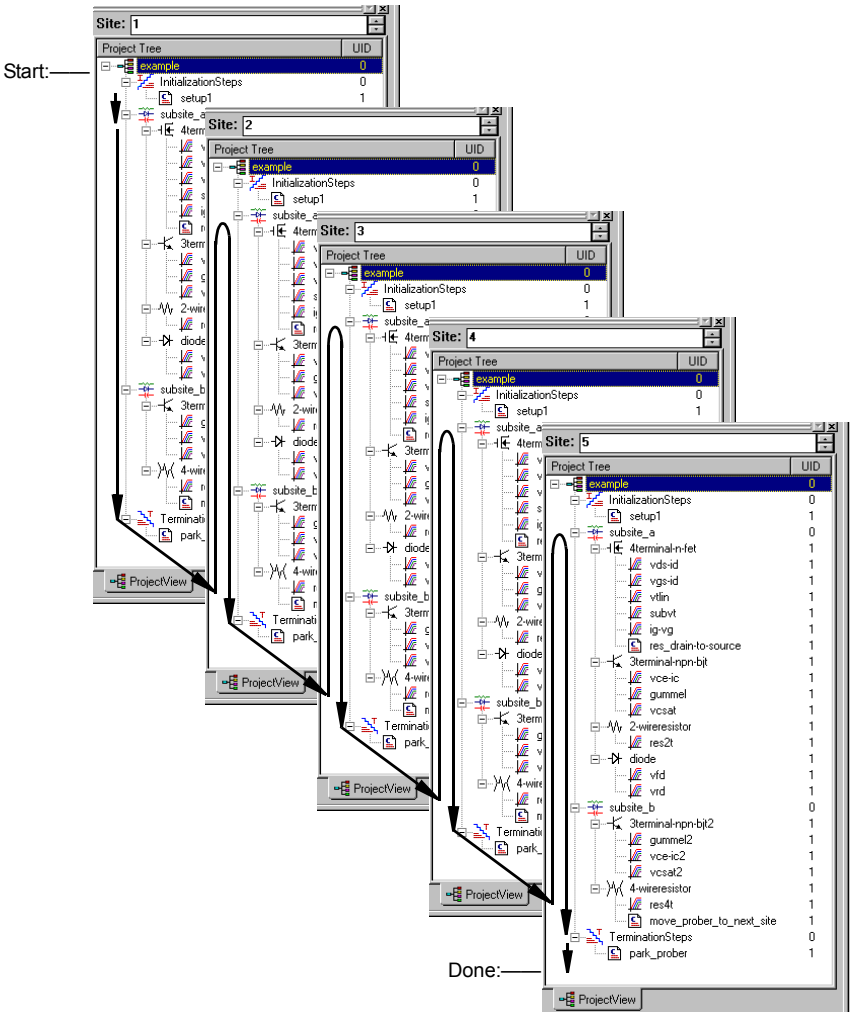


Each time the prober visits a new site, the Test/Plan Indicator box updates and displays the number of the new site.

Figure 6-21 illustrates the following for a Project Plan that visits five sites:

- Initialization of the Project Plan.
- Movement through the site plan.
- Repetition of the site plan each time it reaches **move_prober_to_next_site**, for a total of five repetitions.
- Update of the site number, in the Site Navigator, after each site plan repetition.
- Termination of the Project Plan after the fifth site plan repetition.

Figure 6-21
Movement through the example Project Plan and repetition of the site plan

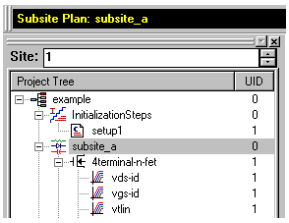


Subsite Plan

A Subsite Plan is a collection of Device Plans and their associated tests. Figure 6-22 highlights how a Subsite Plan is displayed in the Project Navigator. Note that the name of the selected Subsite Plan is also displayed at the top of the Project Navigator in the Test/Plan Indicator box.

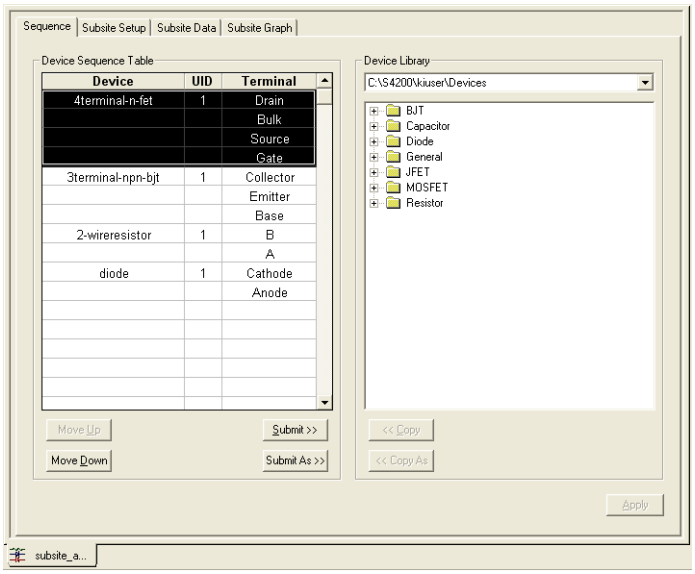
UTMs are required to automatically initiate prober movement between subsites and close matrix channels between devices.

Figure 6-22
Subsite Plan example in Project Navigator



A Subsite Plan is associated with a Subsite Plan window, which facilitates adding, removing, and rearranging subsite Device Plans (described in the next subsection). A Subsite Plan window also allows you to submit Device Plans to a library. To open a Subsite Plan window, double-click on the subsite on the Project Navigator. Figure 6-23 shows the Subsite Plan window for **subsite_a** of the example Project Plan.

Figure 6-23
Subsite Plan window



Subsite cycling

NOTE *For details on subsite cycling, see Subsite cycling later in this section.*

Subsite cycling allows you to repeatedly cycle through the subsite tests. There are three modes of subsite cycling: Cycle Mode and Stress/Measure Mode.

Cycle Mode: For the Cycle Mode, the subsite plan is repeated (cycled) a specified number of times. The collected data for each cycle are stored in the data Sheet tabs for the individual ITMs and UTMs. Measured readings (called Output Values) from individual ITMs and UTMs can be exported into the Subsite Data sheet tab and graphed in the Subsite Graph tab (Output Values versus cycle index).

Stress/Measure Mode: The Stress/Measure Mode integrates stressing with subsite cycling for testing. The first cycle is stress-free. For each subsequent cycle, the devices in the subsite plan are stressed with voltage or current for a specified period of time. After the stress period expires, the tests in the subsite plan are run.

Like the Cycle Mode, Output Values can be exported from ITMs and UTMs into the Subsite Data sheet. Listed in the data sheet is the % Change between each post-stress reading and the corresponding pre-stress reading. For device degradation evaluation, Output Values can be assigned as Targets (in %). When all the targets for a device are reached, then that device will no longer be tested in subsequent cycles. When using Targets, the subsite plan will abort when all Targets are reached or the specified number of cycles are completed.

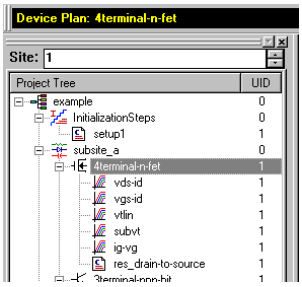
Graphs for collected Subsite Plan data are plotted in the Subsite Graph tab (Degradation versus stress time).

Segment Stress/Measure Mode: This mode is similar to the standard Stress/Measure Mode, but is performed using Segment ARB® waveforms for stressing.

Device Plan

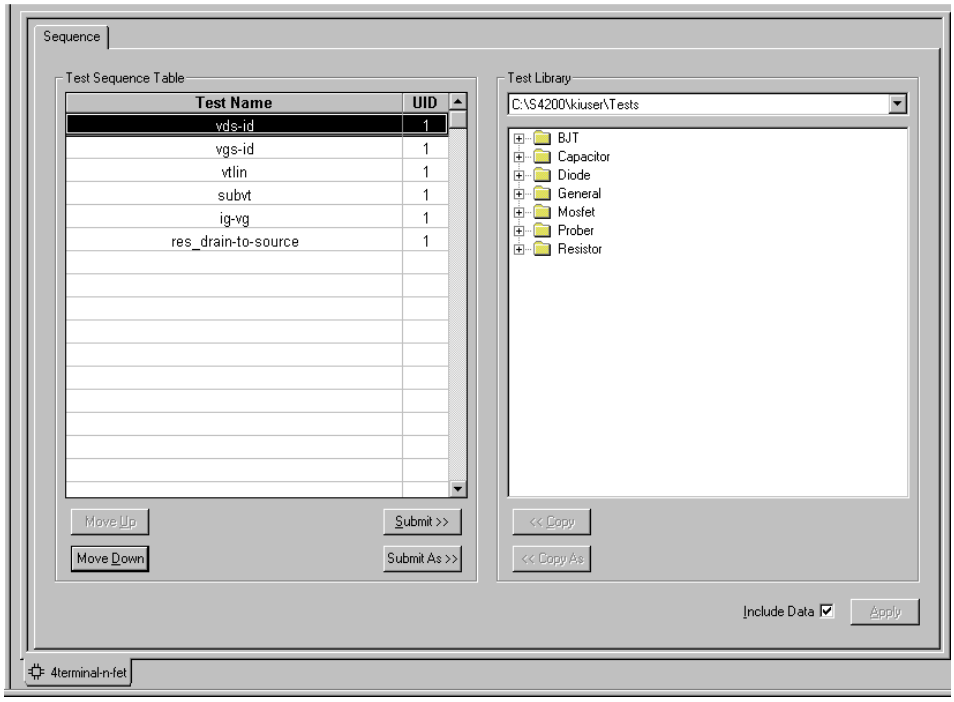
A Device Plan includes the name of a device and the collection of tests to be performed on the device. Figure 6-24 highlights how a Device Plan is displayed in the Project Navigator. Note that the name of the selected Device Plan is also displayed at the top of the Project Navigator in the Test/Plan Indicator box.

Figure 6-24
Device Plan example in Project Navigator



A Device Plan is associated with a Device Plan window, which facilitates adding, removing, and rearranging of ITMs and UTMs (described in the next two subsections). A Device Plan window also allows you to submit ITMs and UTMs to a library. To open a Device Plan window, double-click on the subsite on the Project Navigator. Figure 6-25 shows the Device Plan window for the **4terminal-n-fet** device of the example Project Plan.

Figure 6-25
Device Plan window



Interactive test module (ITM)

ITMs are parametric tests that are easily configured by using a friendly graphical user interface. Double-clicking a typical ITM displays a schematic of the device to be tested and, at each terminal, an instrument object appears. No programming is necessary. KITE comes with a library of ITMs for commonly used devices, including transistors, diodes, resistors, capacitors, and so on. However, within the constraints of the available forcing and measurement functions, existing ITMs may be modified without programming to create a large variety of custom tests.

In Figure 6-26, one of the ITMs of the example Project Plan is shown selected. Note that the name of the selected ITM is also displayed at the top of the Project Navigator in the Test/Plan Indicator box.

Figure 6-26
ITM (Interactive Test Module) displayed in Project Navigator

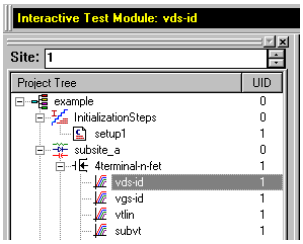
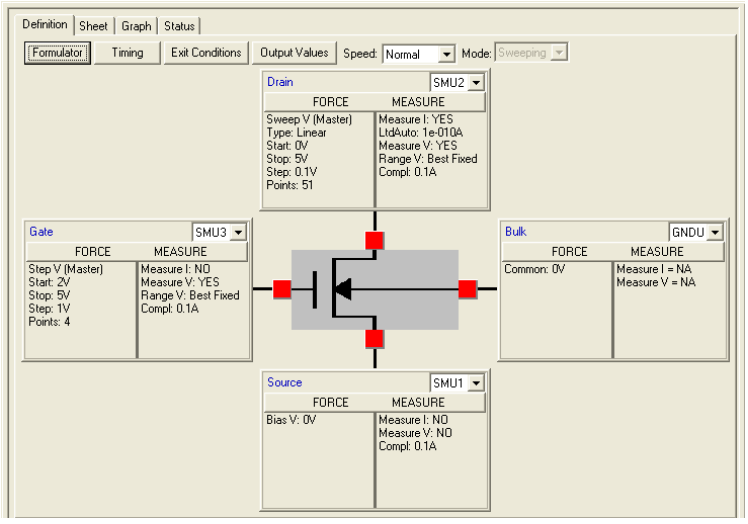


Figure 6-27 displays a typical ITM **Definition** tab that opens when an ITM in the Project Navigator is double-clicked. An ITM **Definition** tab interfaces the user to a variety of configuration tools. Its features include the following:

- Provides for internal instrument selection for each device terminal.
- Displays test settings, for each device terminal.
- Leads to a variety of windows, screens, menus, and so on used to configure forcing and measurement functions, test timing, data filtering, and data analysis.

The **Definition** tab is just one of four primary interfaces available for each ITM (along with several secondary interfaces). The other primary interfaces are the **Sheet**, **Graph**, and **Status** tabs. **Sheet** and **Graph** tabs allow you to configure and display tabular and graphical test results. A **Status** tab reports the test's configuration status.

Figure 6-27
Typical ITM window, which accesses the Definition, Sheet, Graph, and Status tabs



For detailed information about defining and configuring ITMs, refer to Configuring the Project Plan ITMs later in this section.

User test module (UTM)

A UTM is a user-named test module that connects to, configures, and runs a KULT created user module: a dynamic link library (DLL). UTMs may be used to run special parametric tests that cannot be performed with existing ITM functions. Additionally, UTMs may be used to manipulate instrumentation that is external to the Model 4200-SCS. For example, a prober, a C-V meter, a pulse generator, or a switch matrix can be manipulated using a UTM. A UTM can be inserted into a Project Plan in much the same way as an ITM.

A completely new UTM is created by first inserting it into a Project Plan as a name. Then, through UTM **Definition** tab, the UTM is connected to a user module and then configured. A UTM **Definition** tab allows you to configure certain module input parameters.

Data generated by a UTM is displayed in its own **Sheet** and **Graph** tabs. UTM **Sheet** and **Graph** tabs have the same features and characteristics as ITM **Sheet** and **Graph** tabs.

KITE includes user libraries containing precoded user modules for several commonly used external instruments. Additionally, using KULT, you may program custom user-modules in C. KULT includes a library of C-functions that are specially designed for parametric-tests: the Linear

Parametric Test Library (LPTLib). However, any C routine that can be compiled using KULT may be used as source code for a user module.

Figure 6-28 shows the Project Plan positions of two types of UTMs. The first, **res_drain-to-source**, is a parametric test that measures FET drain-to-source resistance at saturation. The second, **move_prober_ to_next_site**, signals a prober to move across a wafer to the next site to be evaluated. Note that the Project Navigator displays a special icon next to each UTM, to differentiate it from an ITM. Also note that the name of the selected UTM is displayed at the top of the Project Navigator in the Test/Plan Indicator box.

Figure 6-28
UTMs (User Test Modules) displayed in Project Navigator

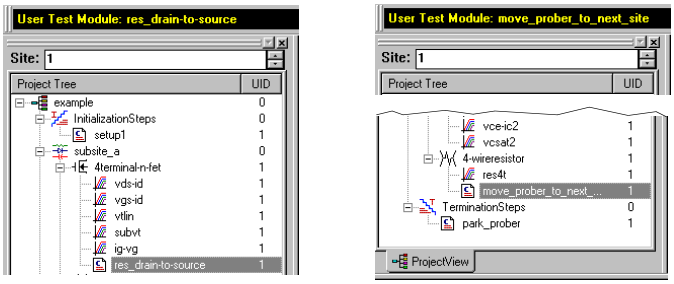
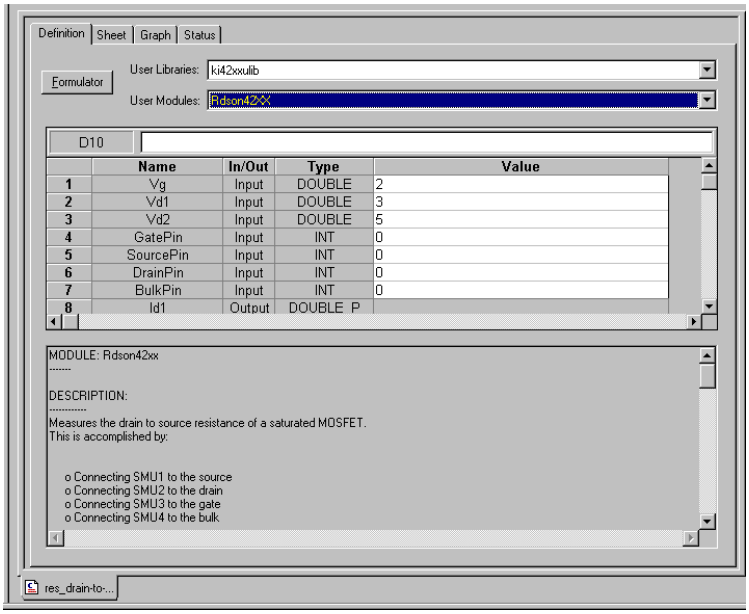


Figure 6-29 illustrates the **Definition** tab for the **res_drain-to-source** UTM in the example Project Plan. Here the UTM **res_drain-to-source** has been created by inserting **res_drain-to-source** as a name in the Project Navigator and then connecting it to (associating it with) the **Rdson42xx** user module. The **Rdson42xx** user module is a KITE supplied user module that is located in the **KI42xxulib** user library provided by Keithley Instruments.

Figure 6-29
Definition tab of a typical UTM window



For detailed information about connecting UTMs to user modules, refer to Configuring the UTMs later in this section. For detailed information about modifying and managing user modules and managing user libraries, refer to Keithley User Library Tool (KULT) in Section 8.

Building, modifying, and deleting a Project Plan

A Project Plan, such as the example Project Plan used to illustrate concepts in previous subsections, is created from a series of building blocks. Most of these building blocks are available in Keithley-supplied libraries. However, if custom ITMs are needed, they can be created as new ITMs or by modifying existing ITMs. Custom user libraries can be created using KULT.

This manual approaches Project Plan creation as follows:

1. Building the entire Project Plan structure without regard to ITM and UTM configuration, as discussed in the following paragraphs.
2. Configuring the ITMs and UTMs after the Project Plan structure is complete, as discussed in Configuring the Project Plan ITMs and Configuring the UTMs later in this section.

However, ITMs and UTMs may be configured at any time.

NOTE *The pulse generator card and scope card are not supported by ITMs at this time. UTMs must be used to control the pulse generator cards and scope card. Alternatively, the new KPulse and KScope applications can be used to interactively control the pulse generator cards and scope card, respectively.*

You can build a completely new Project Plan (from scratch) or modify an existing Project Plan. Both approaches are presented here, in the following order:

- Building a completely new Project Plan is presented first. Keithley Instruments recommends reading this section first, because: 1) it systematically covers the insertion of all Project Plan components; and 2) best relates to the Project Plan structure just discussed.
- Modifying an existing Project Plan is presented next. Keithley Instruments recommends reading this section second, because most of the needed principles discussed in Building a completely new Project Plan are not repeated. However, the modification approach is ultimately easier and faster if much of an existing Project Plan can be reused.

Occasionally, you may wish to delete a project plan. Deleting a Project Plan describes how to delete a project plan without using Windows Explorer.

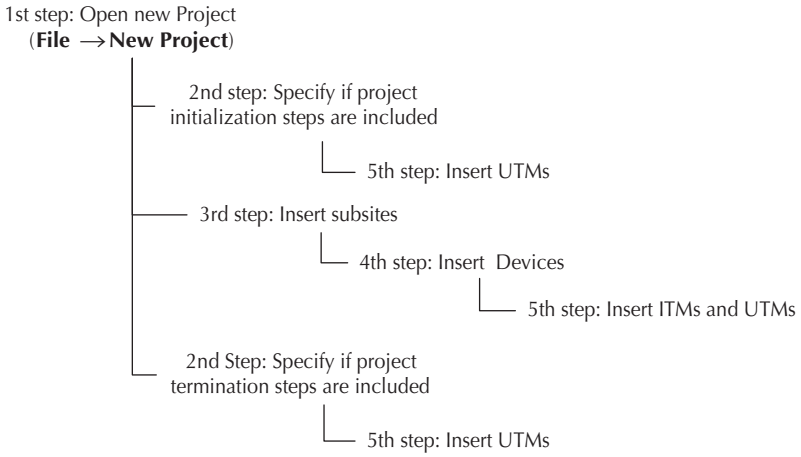
Building a completely new Project Plan

This section leads you through building a new Project Plan in the following order, as illustrated in Figure 6-30:

1. Specifying initialization and termination steps, if needed.
2. Inserting all the Subsite Plans.
3. Inserting all the Device Plans.
4. Inserting all of the ITMs.
5. Inserting all of the UTMs.

The construction of an example Project Plan called **“u_build”** is used periodically to illustrate the process.

Figure 6-30
Hierarchical Project Plan construction



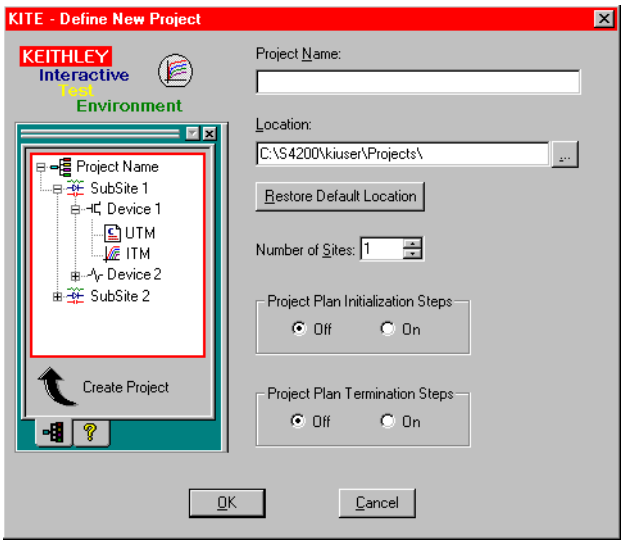
The topic headings for this subsection are as follows:

- Defining the new Project Plan
- Inserting the Subsite Plans
- Inserting Device Plans
- Inserting the ITMs
- Inserting the UTMs
- Saving the Project Plan

Defining the new Project Plan

1. In the File menu, click **New Project**. The Define New Project window opens, with the default settings as shown in Figure 6-31.

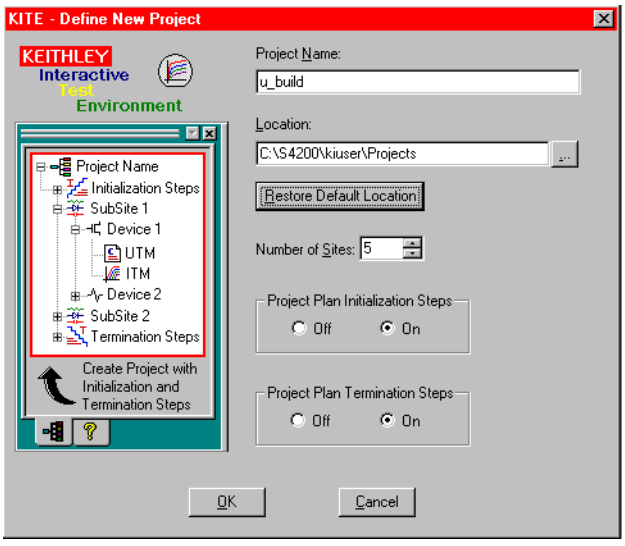
Figure 6-31
Define New Project window



2. Configure the Define New Project window to meet the needs of your Project Plan, as follows:
   - **Project Name** edit box: Enter the Project Plan name, subject to the following rules.
     - Any combination of alphanumerical characters, dashes, and underscores is allowed, but no other special characters are allowed.
     - No spaces are allowed.
     - 260 characters, maximum, is allowed for the following *combination*: Project Plan name *plus* all characters used in any directory path that includes the Project Plan name.
   - **Location** edit box: Specify the folder where the Project Plan is to be stored. Keithley Configuration Utility (KCON).
   - **Number of Sites** combo box: Specify the number of sites to be evaluated by the Project Plan, either by typing in the number of sites or by using the spin buttons.
   - **Restore Default Location** button: If you change the Project Plan location, click this button if you wish to restore the default folder as the Project Plan location.
   - **Project Plan Initialization Steps:** Click **On** if you plan to insert one or more initialization UTMs at the start of the Project Plan.
   - **Project Plan Termination Steps:** Click **On** if you plan to insert one or more termination UTMs at the end of the Project Plan.

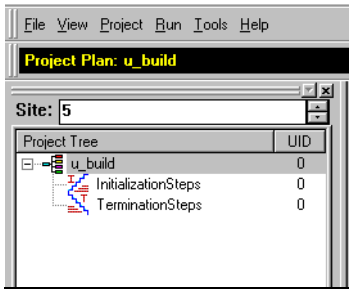
   Figure 6-32 shows the **u_build** Project Plan being configured to start with **Project Plan Initialization Steps**, to evaluate five sites, and to conclude with **Project Plan Termination Steps.** Note that place holders for initialization and termination steps have been added to a mini-Project Navigator display at the left of the window, and that the **Create Project** caption has changed to **Create Project with Initialization and Termination Steps**.

Figure 6-32
Define New Project window configured for the u_build Project Plan



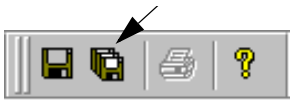
3. Click **OK.** The Project Navigator appears, reflecting the chosen configuration. See Figure 6-33.

Figure 6-33
Initial Project Navigator window for the u-build Project Plan



NOTE *After you configure the Define New Project window, the Project Plan, essentially empty, exists in the selected file location. As you proceed with the construction, you can save the Project Plan at any time by clicking the* ***Save All*** *toolbar button.*

To close the Project Plan at any time, select ***Close Project*** *in the* ***File*** *menu. If you did not save the Project Plan, a dialog box prompting you to save will appear; click* ***Yes*** *to save.*

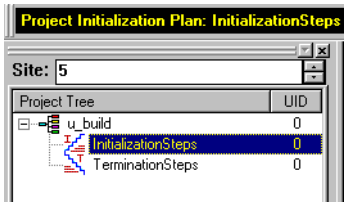


If you change your mind about a Project Plan component that you insert, you can delete the component by selecting it and pressing the ***DELETE*** *key on the keyboard. Keep in mind, however, that deleting a Device Plan simultaneously deletes all of its tests and that deleting a Subsite Plan simultaneously deletes all of its Device Plans and tests.*

Inserting the Subsite Plans

1. In the Project Navigator, select the Project Plan component below to insert the first Subsite Plan. For the **u_build** Project Plan, the appropriate component to select is **Initialization Steps**. See Figure 6-34.

Figure 6-34
Selecting where the first Subsite Plan should go

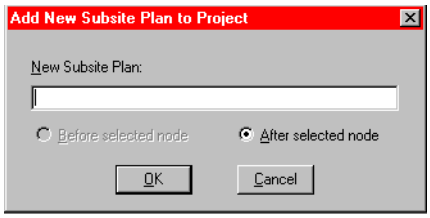


2. Add a Subsite Plan to the Project Plan as follows:
   a. Do either of the following:
      - In the **Project** menu, click **New Subsite Plan.**
      - On the Project Plan toolbar, click the **Add Subsite Plan** button. See Figure 6-35.

Figure 6-35
Add Subsite Plan button

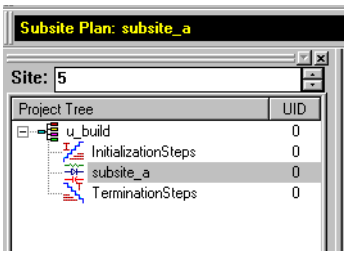
The Add New Subsite Plan to Project dialog box appears. See Figure 6-36.

Figure 6-36
Add New Subsite Plan to Project dialog box



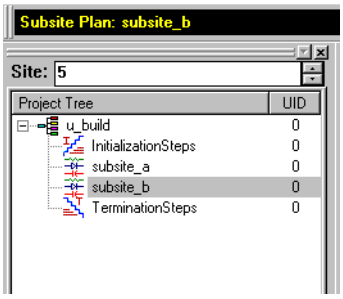
   b. Enter the name for the new Subsite Plan, subject to the following rules:
      - Any combination of alphanumerical characters, dashes, and underscores is allowed, but no other special characters are allowed.
      - No spaces are allowed.
      - 260 characters, maximum, is allowed for the following combination: Subsite Plan name *plus* all characters used in any directory path that includes the Subsite Plan name.
   c. Click **OK**. The Subsite Plan is inserted below the selected component. See Figure 6-37.

Figure 6-37
Inserted Subsite Plan



3. Leaving the added Subsite Plan selected, insert additional Subsite Plans as needed by repeating steps 1 and 2. Figure 6-38 shows two Subsite Plans, total, inserted into the **u_build** Project Plan.

Figure 6-38
Completed Subsite Plan insertions



NOTE *It is often most convenient to insert components into a new Project Plan sequentially, from top to bottom by 1) selecting an existing component just above where you want the new component and 2) inserting the component using the default **After selected node** option. This procedure generally reflects that approach. However, where appropriate, KITE also allows you to insert a new component before a selected existing component, using the **Before selected node** option.*

Inserting Device Plans

You insert a Device Plan into your Project Plan from the default device library in `C:\4200\kiuser\Devices` or an equivalent personal device library such as `C:\4200\YourName\Devices`. The devices available in the library include the standard set of devices that come installed on the Model 4200-SCS, as well as any custom-name devices that you have submitted (the submittal procedure is discussed under Submitting devices to a library later in this section).

This section describes four scenarios for inserting Device Plans:

- Inserting Device Plans using the default library name
- Inserting a Device Plan using a new name
- Inserting multiple instances of a Device Plan using the same name
- Inserting multiple instances of a Device Plan using different names

For illustration purposes, the same type of Device Plan (for a four-terminal n-FET) is inserted in all scenarios. However, the instructions apply equally to all Device Plans.

NOTE *You cannot insert Device Plans under **Initialization Steps** and **Termination Steps** in the Project Navigator.*

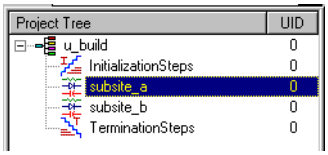
Inserting Device Plans using the default library name

NOTE *You can insert and rearrange Device Plans from the device library using the Subsite Plan Window (refer to Adding, rearranging, and deleting Device Plans). However, the methods described below are typically easier and faster when building completely new Project Plans.*

Insert a Device Plan using the default name, as follows:

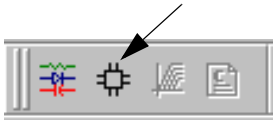
1. In the Project Navigator, select the Subsite Plan component below to insert the first device. To add the *first* Device Plan to **subsite_a** of the **u_build** Project Plan, the appropriate component to select is **subsite_a**. See Figure 6-39.

Figure 6-39
Selecting the location for the first library Device Plan in a Subsite Plan



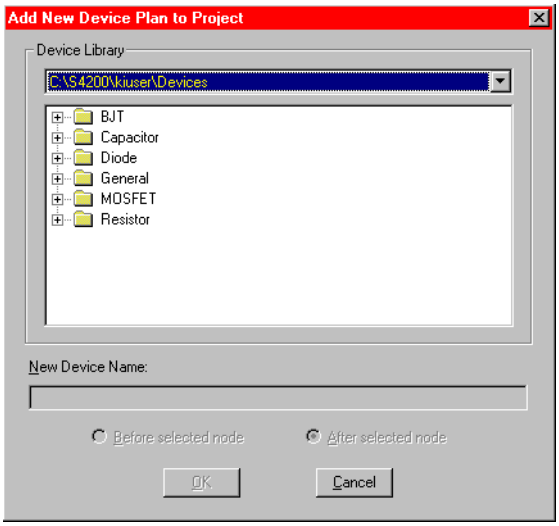
2. Add the Device Plan to the Project Plan as follows:
   a. Do either of the following:
      - In the **Project** menu, click **New Device Plan**.
      - In the Project Plan toolbar, click the **Add New Device Plan** button as shown in Figure 6-40.

Figure 6-40
Add New Device Plan button



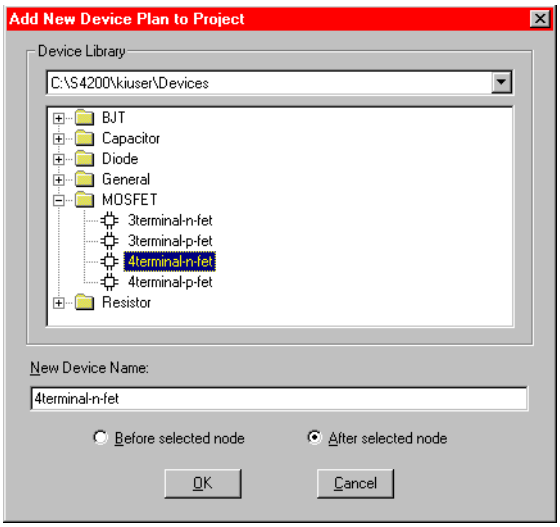
- The Add New Device Plan to Project window appears. See Figure 6-41.

Figure 6-41
Add New Device Plan to Project window



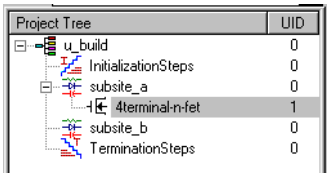
b. In the Add New Device Plan to Project window, select a new Device Plan from the default device library. For the **u_build** Project Plan, the first Device Plan to be added is a MOSFET Device Plan called **4terminal-n-fet**. See Figure 6-42.

Figure 6-42
Selecting a Device Plan



c. Click **OK**. The Device Plan is inserted below the selected component. See Figure 6-43.

Figure 6-43
Device Plan inserted using default name



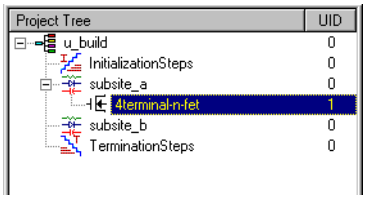
Inserting a Device Plan using a new name

You may insert any Device Plan anywhere in the same Project Plan under a new name. To insert a Device Plan using a new name, do the following:

1. Follow steps 1, 2a, and 2b as instructed just above under Inserting Device Plans using the default library name earlier in this section.

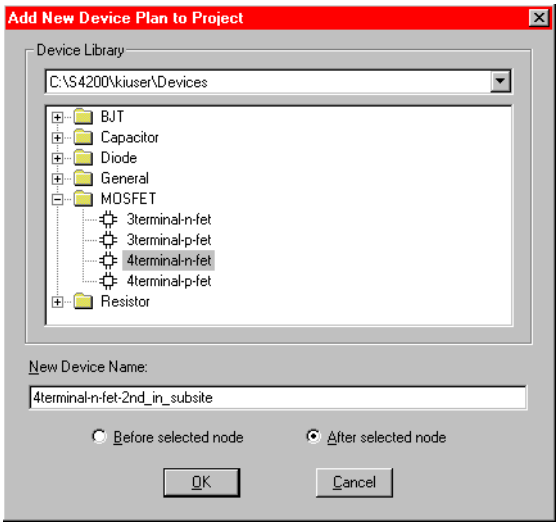
   To add a second Device Plan below an existing Device Plan, select the existing Device Plan name in the Project Navigator. For example, to add a new Device Plan below **4terminal-n-fet** in **subsite_a** of the **u_build** Project Plan, select **4terminal-n-fet**. See Figure 6-44.

Figure 6-44
Selecting locations for the 2nd, 3rd, and so on. Device Plans in a Subsite Plan



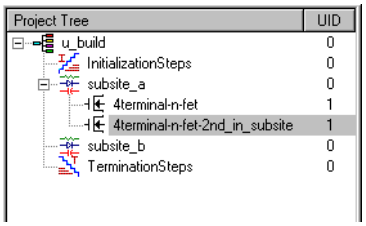
2. When the Add New Device Plan to Project window opens, do the following:
   a. Select the device to be added. For the **u_build** Project Plan, **4terminal-n-fet** was selected again for insertion in **subsite_a**, to illustrate that multiple instances of a given device are permitted anywhere in the Project Plan *if each instance is given a different name*.
   b. In the **New Device Name** edit box, replace the default name with a new name. For the **u_build** Project Plan, the name **4terminal-n-fet** was replaced with **4terminal-n-fet_2nd_in_subsite**. See Figure 6-45.

Figure 6-45
Selecting and renaming a library Device Plan



3. Click **OK**. The Device Plan is inserted below the selected component. See Figure 6-46.

Figure 6-46
Device Plan inserted using a new name



Inserting multiple instances of a Device Plan using the same name

You may not insert multiple instances of a Device Plan with the same name in the same Subsite Plan. However, by using device library management you may insert additional instances of a library Device Plan using the same name in different Subsite Plans, as follows:

In the Project Navigator, double-click the Subsite Plan to open it. From the Sequence tab (see Figure 6-47) copy the desired device into the Subsite Plan. Figure 6-47 shows the three steps to copy a second instance of the **4terminal-n-fet** into the Project Plan. Figure 6-48 shows the **4terminal-n-fet** added to the subsite_b Subsite Plan.

Figure 6-47
Using device library management to add a Device Plan

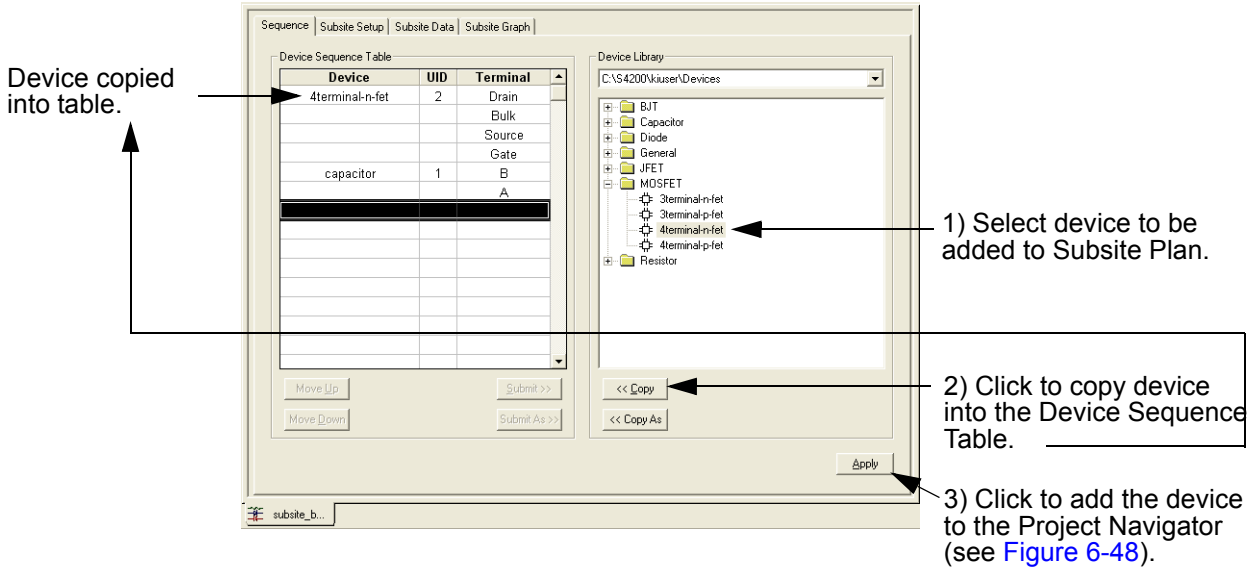
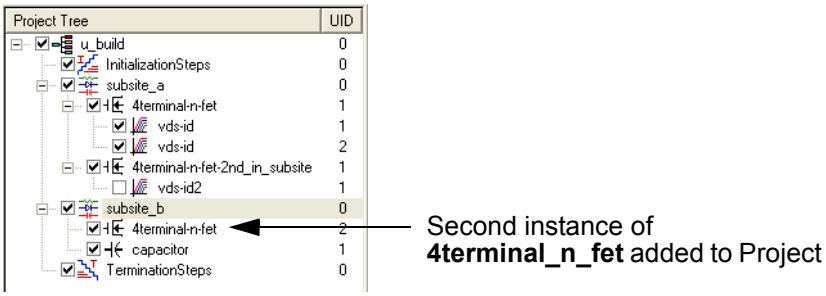


Figure 6-48
Device Plan inserted into Project Navigator



Inserting multiple instances of a Device Plan using different names

If you use a new Device Plan name each time, you may insert multiple instances of a Device Plan anywhere in the same Project Plan, without restriction. Refer to the procedure above, Inserting a Device Plan using a new name earlier in this section.

Editing the Device Plan insertions

For information on editing your Device Plan insertions, refer to Adding, rearranging, and deleting Device Plans later in this section.

Inserting the ITMs

The next step, after inserting Device Plans into your Project Plan, is to insert the ITMs and UTMs. This manual discusses and illustrates ITM insertion before UTM insertion. However, depending on your needs and preferences, the reverse order of insertion (or a mixed order of insertion) may be advantageous.

You can insert an ITM into your Project Plan in two ways:

- **From scratch**: as a completely new, unconfigured ITM to be customized for your requirements.
- **From a test library**: as a previously defined and configured ITM, either to be used essentially as-is or to be customized for your requirements.

Both ways are discussed in the following paragraphs.

CAUTION **If you need to insert multiple instances of an ITM under the same name, be sure to first read Inserting multiple instances of a library ITM using the same name later in this section. Also review Table 6-3 below.**

Table 6-3
Shared and unique characteristics for same-named Project Plan ITMs

Characteristics that are shared between all instances of a Project Plan ITM having the same name. A change of one instance changes the definition of *all* instances identically.*	**Characteristics that are unique to each instance of a Project Plan ITM having the same name. A change of one instance has no effect on any other instance.**
Everything on the **Definition** tab for the ITM, including the following: • The instrument selections for each instrument object • The instrument settings for each selected instrument (the configurations implemented using the Forcing-Function/Measure-Options windows for all device terminals) • All **Formulator** formulas* • All **Timing** settings • **Exit Conditions** • **Output Values** • The **Speed** selection • The **Mode** selection	• Test data • Formulas in the **Calc** worksheet of the **Sheet** tab • Plot settings in the **Graph** tab • The Unique IDentifier number (**UID**)

* Some changes to an ITM result in the deletion of all **Formulator** formulas.

NOTE *An ITM can be inserted into a KITE Project Plan and attached to any target device, subject to the following rules:*

- *The number of device terminals required by the ITM must not be greater than the number of terminals of the target device.*
- *Each terminal name required by the ITM must be present on the target device.*

*You cannot insert ITMs under **Initialization Steps** and **Termination Steps** in the Project Navigator.*

Inserting ITMs from a test library

Insert a library ITM into your Project Plan from the default test library in `C:\4200\kiuser\Tests` or an equivalent personal test library such as `C:\4200\YourName\Tests`. Such a library typically includes the standard set of ITMs that come installed on the Model 4200-SCS, as well as any custom ITMs that you have submitted (the submittal procedure is discussed under Submitting tests to a library later in this section).

Factory-supplied library ITMs, if unmodified, are preconfigured with commonly used parameter settings and are usually accompanied by a set of typical data for illustration purposes. Custom library ITMs are also often preconfigured, though for special requirements. When appropriate, inserting a library ITM (and then later reconfiguring it if necessary) is often the most efficient way to add an ITM to your Project Plan.

This section describes four scenarios for inserting library ITMs:

- Inserting a library ITM using the "default" library name
- Inserting a library ITM using a new name
- Inserting multiple instances of a library ITM using the same name
- Inserting multiple instances of a library ITM using a different name

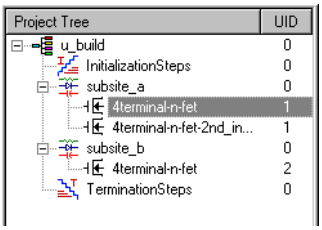
For illustration purposes, the same type of ITM (the **"vds-id"** ITM) is inserted in all scenarios. However, the instructions apply equally to all ITMs.

Inserting a library ITM using the "default" library name

To insert an ITM from a library using the default name, do the following:

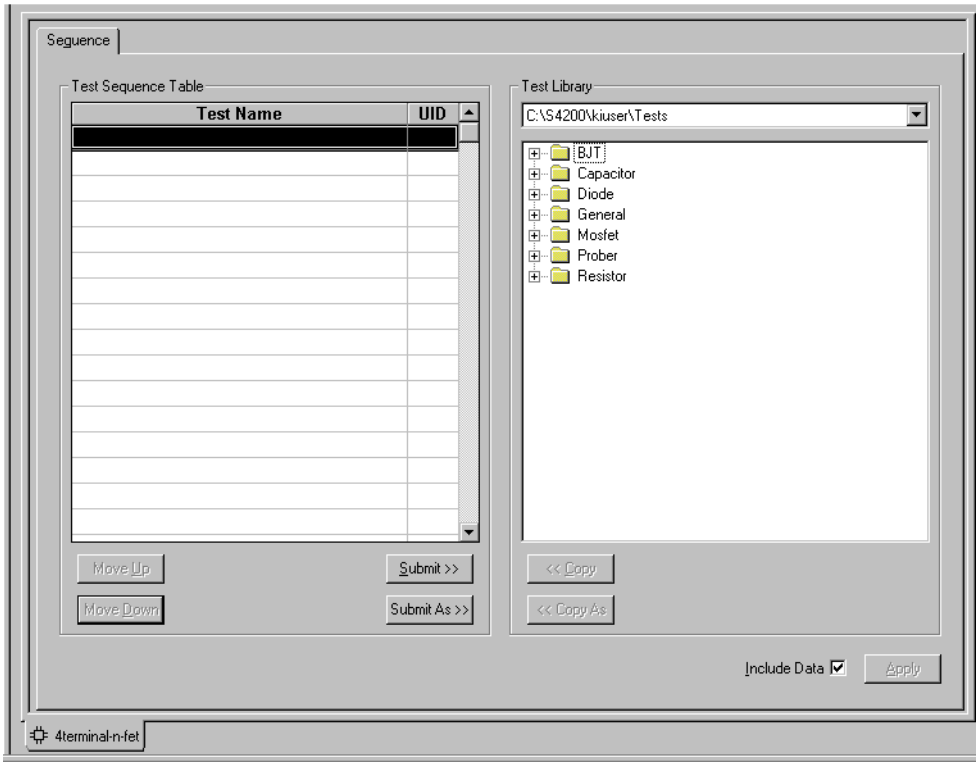
1. In the Project Navigator, select the Device Plan component below to insert the first ITM. To add the first ITM to **subsite_a** of the **u_build** Project Plan, an appropriate component to select is **4terminal-n-fet.** See Figure 6-49.

Figure 6-49
Selecting the location in the device for the first library ITM



2. Double-click the Device Plan name in the Project Navigator. The Device Plan window appears, displaying a tree of device category folders. Each folder contains a list of device-appropriate ITMs. See Figure 6-50.

Figure 6-50
Device Plan Window

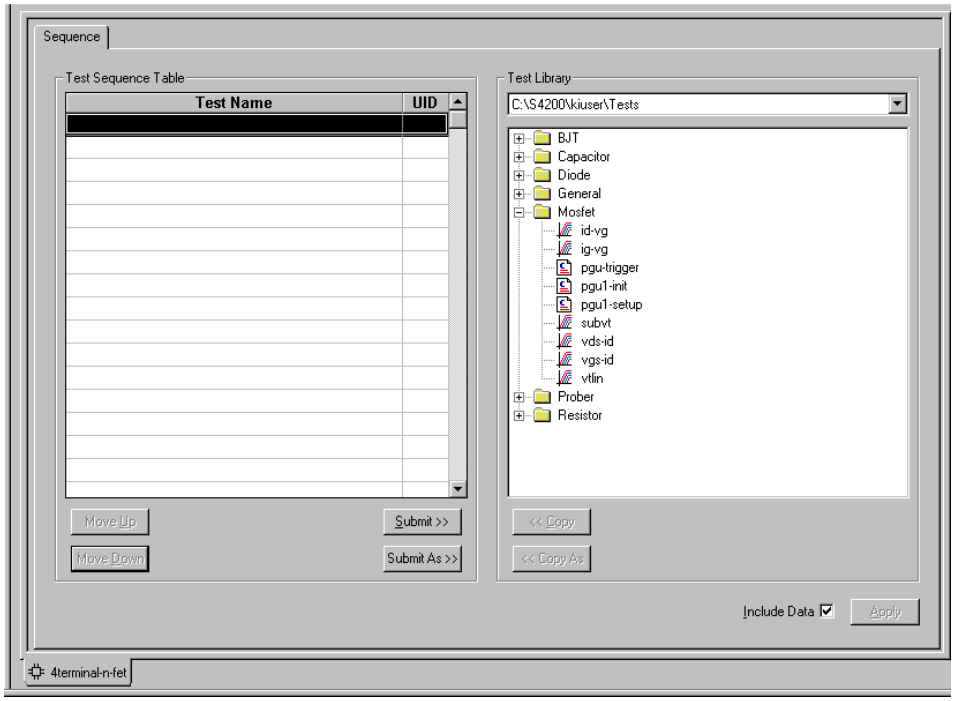


3. In the **Test Libraries** combo box, select the test library that contains the desired ITM. The default test library is `C:\S4200\kiuser\Tests`, unless another library, such as a `C:\S4200\YourName\Tests` personal library was chosen as the default (using KCON, as described in Changing the active user-library directory in Section 8).

NOTE *If no other selections are present in the **Test Libraries** combo box, additional libraries can be added through the **Directories** tab of the **Tools > Options** menu. Refer to Customizing KITE later in this section).*

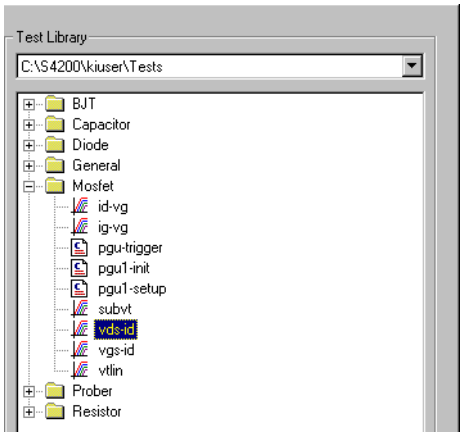
4. Click the folder for the device type under which you are adding the ITM. A list of ITMs (and, typically, UTMs) appears. Figure 6-51 shows the MOSFET ITM selections that were available for insertion into the **4terminal-n-FET** Device Plan of the **u_build** Project Plan.

Figure 6-51
Example of ITMs listed under a device type



5. Select the desired test. Figure 6-52 illustrates selection of the **"vds_id"** ITM for the **u_build** Project Plan.

Figure 6-52
Selecting an ITM



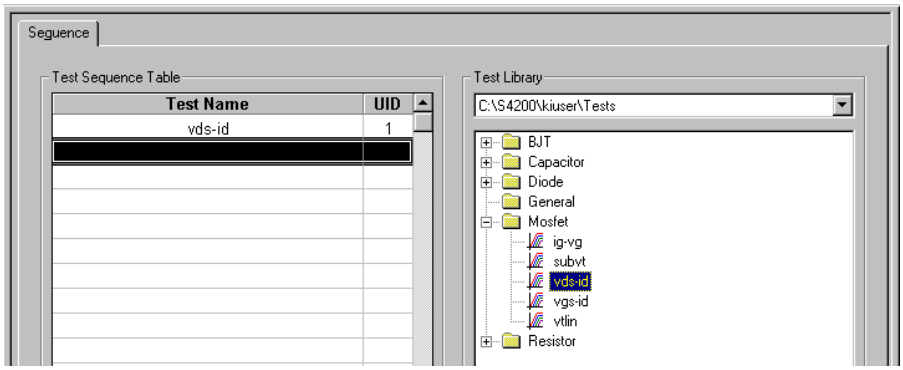
6. Many ITMs contain sample data. If you wish to include this data when you insert the ITM, make sure that the **Include Data** checkbox is checked.

Figure 6-53
Include Data check box

The **Include Data** checkbox is located in the lower right corner of the Device Plan window.

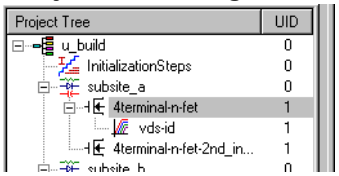
7. In the Device Plan window, below the list of ITMs, click the **Copy** button. The ITM is added to the **Test Sequence Table** of the Device Plan window. See Figure 6-54.

Figure 6-54
Adding an ITM to the Test Sequence Table



8. If, under **Test Sequence Table**, an ITM is not at the preferred position in the sequence, do this:
   a. Select the ITM to be moved.
   b. Use the **Move Up** button or the **Move Down** button to reposition the ITM.
9. In the Device Plan window, below the list of ITMs, click the **Apply** button. The ITM is added to the Project Navigator. See Figure 6-55.

Figure 6-55
Library ITM inserted in the Project Plan using the default library name



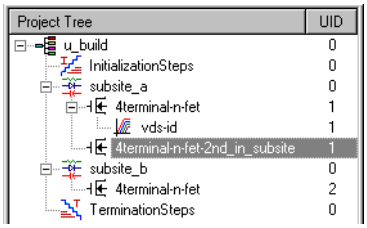
Inserting a library ITM using a new name

You may insert an ITM from the library anywhere in the same Project Plan under a new name. A renamed ITM is independent of the originally named ITM. Therefore, a renamed ITM can be custom-configured to meet the specific test requirements of the device with which it is associated.

To insert a library ITM using a new name, do the following:

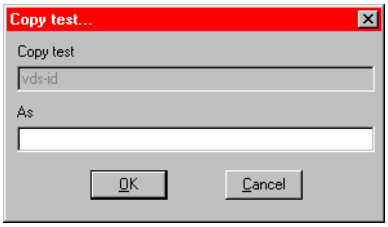
1. Select the Device Plan under which to insert a renamed ITM. For the **u_build** Project Plan, a second, somewhat modified n-FET, the **4terminal-n-fet-2nd_in-subsite**, was selected. See Figure 6-56.

Figure 6-56
Selecting the location for a second "vds-id" ITM for the u_build Project Plan



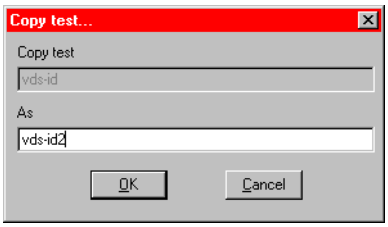
2. Open the Device Plan, select a test library, and select a test as described in steps 2 through 6 of Inserting a library ITM using the "default" library name earlier in this section. For the **u_build** Project Plan, the **"vds-id"** ITM was selected from the default test library, `C:\S4200\kiuser\Tests`.
3. In the Device Plan window, below the list of ITMs, click the **Copy As** button. The **Copy Test** dialog box appears. See Figure 6-57.

Figure 6-57
Copy Test dialog box



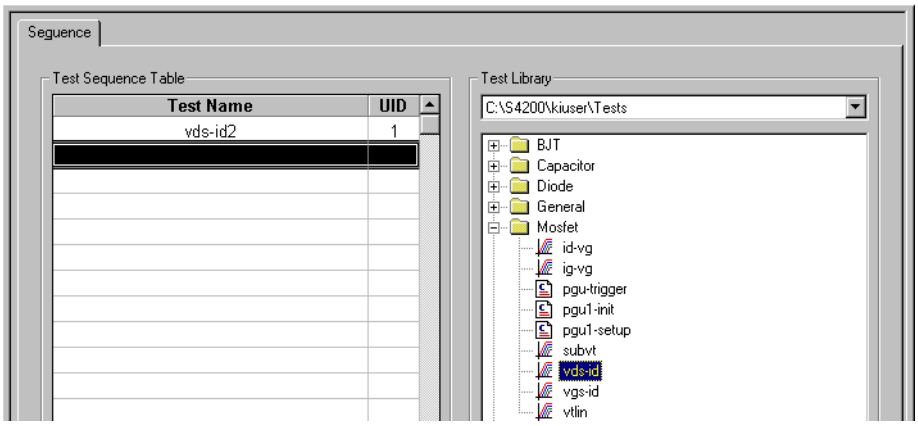
4. In the **Copy Test** dialog box, enter the new name for the ITM. For the **u-build** Project Plan, the new name **"vds-id2"** was entered. See Figure 6-58.

Figure 6-58
New name entered for a library ITM



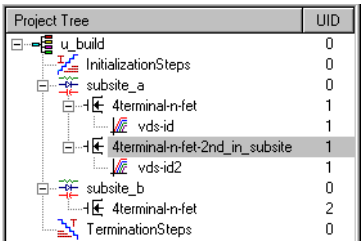
5. Click **OK**. The renamed ITM is added under **Test Sequence Table** in the Device Plan window. See Figure 6-59.

Figure 6-59
Renamed library ITM added to the Test Sequence Table



6. If, under **Test Sequence Table**, an ITM is not at the preferred position in the sequence, do this:
   a. Select the ITM to be moved.
   b. Use the **Move Up** button or the **Move Down** button to reposition the ITM.
7. In the Device Plan window, below the list of ITMs, click the **Apply** button. The renamed ITM is added to the Project Navigator. See Figure 6-60.

Figure 6-60
Renamed KITE library ITM inserted in the Project Plan



Inserting multiple instances of a library ITM using the same name

You may insert multiple instances of any ITM (from any ITM library) anywhere in the Project Plan using the same name. However, in a given Project Plan, KITE *automatically* keeps all same-named ITMs configured identically (the data for each ITM remains independent). Although this feature is very useful, for example, it enables multiple tests to be performed identically at multiple environmental conditions; you must insert same-named ITMs *cautiously*. Before attempting to insert multiple ITMs under the same name, ensure that you understand the following rules (also refer to Table 6-3):

- When you insert a same-named ITM, all of its **Definition** tab settings (including **Formulator** formulas) permanently overwrite the **Definition** tab settings of all existing same-named ITMs in the Project Plan. That is, all existing same-named ITMs take on the configuration of the newly inserted same-named ITM, and *the original configurations are lost*.
- If you change the definition of any one ITM, *all other same-named ITMs in the Project Plan automatically change identically.* Therefore, the **Definition** tabs of all same-named ITMs are always identical.
- However, the data for each same-named ITM is unique. Therefore, the **Sheet** and **Graph** tabs for each same-named ITM are unique.
- Likewise, the Unique IDentifier number (UID) for each same-named ITM is unique. Once assigned, the UID is permanent. If you delete one or more of the same-named ITMs, the UIDs for the remaining same-named ITMs remain the same.

To insert a same-named ITM in a Project Plan from a test library, use the procedure described previously under Inserting a library ITM using the “default” library name earlier in this section. However, when using that procedure, note that when you press **Copy**, a Test Already Exists message box appears, warning you that the ITM that you are about to insert will permanently overwrite all ITMs with the same name. This message appears regardless of where you insert the ITM in the Project Plan, as illustrated by Figures 6-61, 6-62, and 6-63.

Figure 6-61
Result of pressing Copy to add a same-named ITM to a given Device Plan

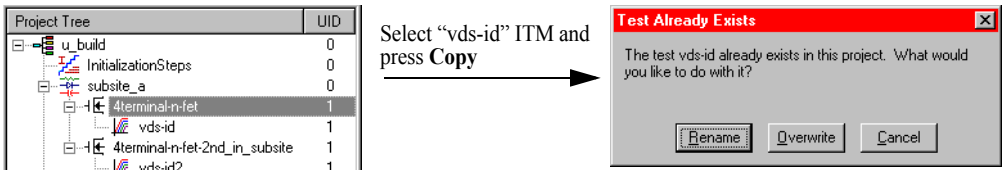


Figure 6-62
Result of pressing Copy to add a same-named ITM within a given Subsite Plan

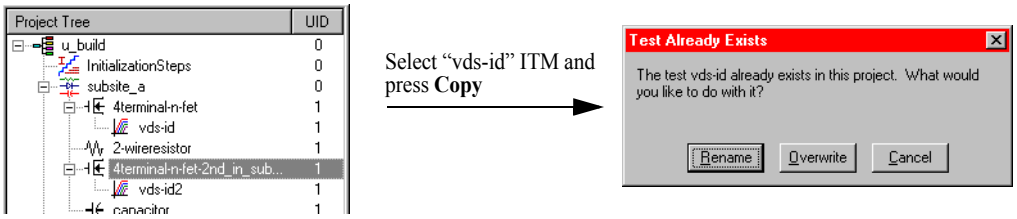
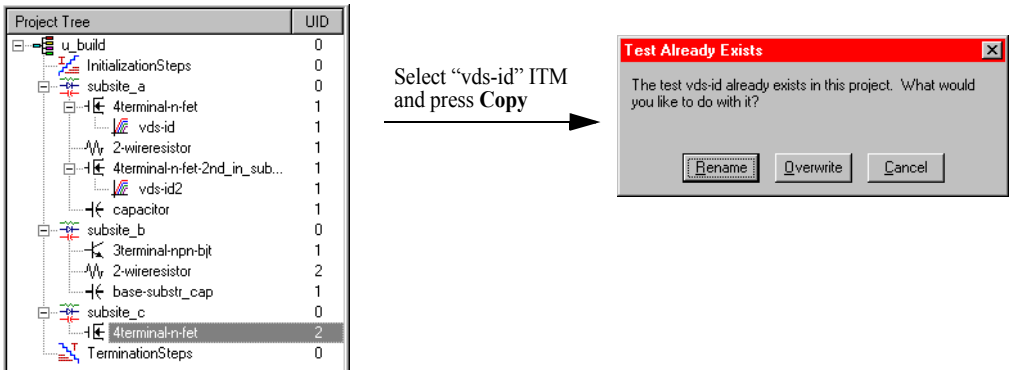


Figure 6-63
Result of pressing Copy to add a same-named ITM to a different Subsite Plan



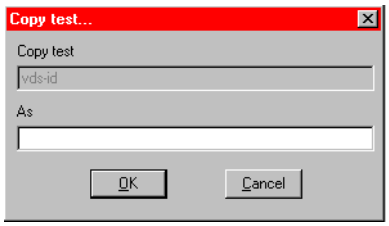
Note that a new Subsite Plan was added to the **u_build** Project Plan, without discussion, for tutorial purposes.

When the Test Already Exists message box appears, do one of the following:

Option I. Rename the ITM: If you do not specifically need a same-named ITM or if the rules for same-named ITM are contrary to your test objectives, *rename* the ITM before inserting it, as follows:

1. Click **Rename** in the Test Already Exists message box. The Copy Test dialog box appears. See Figure 6-64.

Figure 6-64
Copy Test dialog box



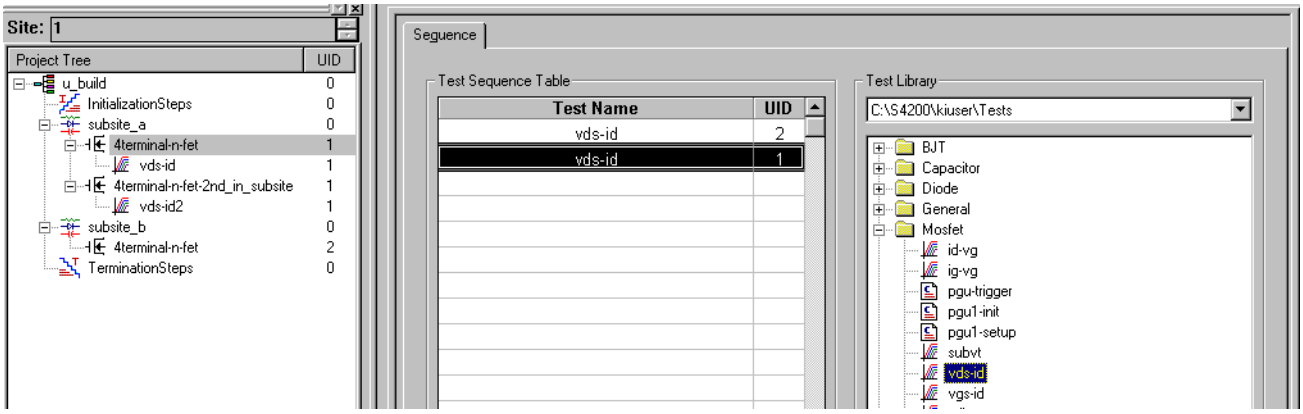
2. Enter a new name.
3. Click **OK**. The new ITM appears under **Test Sequence Table**.
4. If, under **Test Sequence Table**, an ITM is not at the preferred position in the sequence, do this:
   a. Select the ITM to be moved.
   b. Use the **Move Up** button or the **Move Down** button to reposition the ITM.
5. Click **Apply**. The new ITM is added to the Project Plan.

Option II. Insert the ITM as-is: If you specifically need a same-named ITM and the rules for same-named ITM meet your test objectives, insert it without renaming it, as follows:

1. Click **Overwrite** in the Test Already Exists message box. A new instance of the same-named ITM appears under **Test Sequence Table** in the Device Plan window.

   Figure 6-65 shows the **"vds-id"** ITM added to the **Test Sequence Table**, at the location that was selected in Figure 6-61. Note that the next sequential UID number, 2, is assigned to this second-instance of **"vds-id"** in the Project Plan.

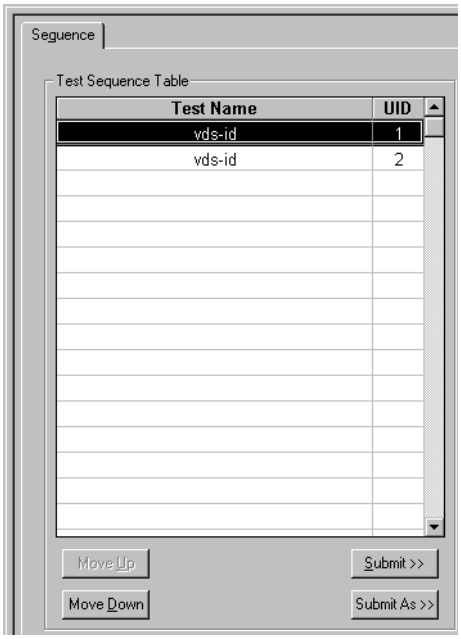
Figure 6-65
Second instance of a same-named ITM added to the Test Sequence Table



2. If, under **Test Sequence Table**, an ITM is not at the preferred position in the sequence, do this:
   a. Select the ITM to be moved.
   b. Use the **Move Up** button or the **Move Down** button to reposition ITM.

   Figure 6-66 shows the results of reversing the order of the two **"vds-id"** ITMs, using **Move Up**, so that their UID numbers are in numerical sequence.

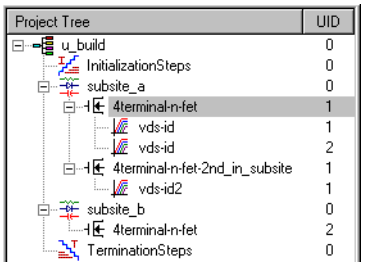
Figure 6-66
Relocating an ITM



3. Click **Apply**. The following occurs:
   - The new ITM is added to the Project Plan.
   - All previously inserted same-named ITMs match the newly inserted ITM (except for their data, **Sheet** tabs, **Graph** tabs, and UID numbers).

   A second instance of the **"vds-id"** ITM was added to the **u-build** Project Plan. See Figure 6-67.

Figure 6-67
Second instance of a same-named ITM added to the Project Plan



Inserting multiple instances of a library ITM using a different name

If you use a new ITM name each time, you may insert multiple instances of an ITM anywhere in the same Project Plan, without restriction. Refer to the procedure above, Inserting a library ITM using a new name.

Inserting a completely new ITM

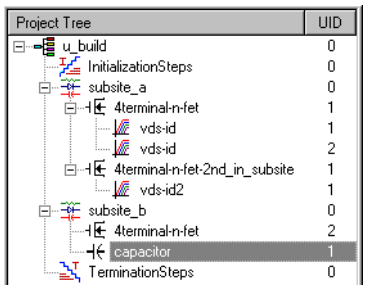
As discussed above, you can insert an ITM directly from a library. However, you can also insert a completely new, unconfigured ITM. This ITM can then be customized and configured to meet special needs, within the ITM-definition constraints for the device to be tested.

NOTE *After inserting a completely new ITM, you cannot insert additional instances of this ITM until you submit it to a test library. For information about submitting an ITM to a test library, refer to Submitting devices, ITMs, and UTMs to libraries later in this section.*

Insert a completely new ITM as follows:

1. Select the device or test below which to add the completely new ITM. For the **u_build** Project Plan, a newly added capacitor in **subsite_b** was selected. See Figure 6-68.

Figure 6-68
Selecting a location in the Device Plan for the completely new ITM

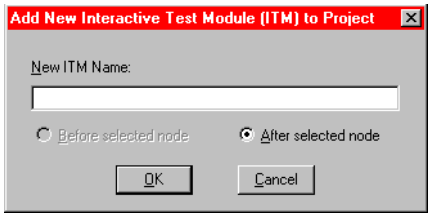


2. Do either of the following:
   - In the **Project** menu, click **New Interactive Test Module.**
   - In the Project Plan toolbar, click the **Add New ITM** button. See below.

Figure 6-69
Add New ITM button

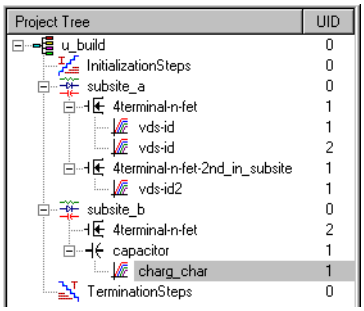
The Add New Interactive Test Module (ITM) to Project dialog box appears. See Figure 6-70.

Figure 6-70
Add New Interactive Test Module (ITM) to Project dialog box



3. In the Add New Interactive Test Module (ITM) to Project dialog box, type the name for the completely new ITM. The name **charg_char** was entered as the new ITM for the capacitor in the **u_build** Project Plan.
4. Click **OK.** The new, as yet unconfigured, ITM is added to the Project Plan. See Figure 6-71.

Figure 6-71
Completely new ITM added to the Project Plan



Editing the ITM insertions

For information on editing your ITM insertions, refer to Rearranging and deleting ITMs and UTMs later in this section.

Inserting the UTMs

In contrast to ITMs, relatively few UTMs are presently provided by Keithley Instruments in the `C:\S4200\kiuser\Tests` library. If these are insufficient, you must initially create additional UTMs (which you can subsequently submit to `C:\S4200\kiuser\Tests` or to a personal library). The following steps summarize the required approach.

1. Insert a new UTM into the Project Plan only by name.
2. Connect the name of the UTM to an existing KULT created user library and user module through a UTM **Definition** tab.
3. In the UTM **Definition** tab, edit/enter the required parameters.

NOTE *Connecting a UTM name with a user library and user module and entering the required parameters is discussed subsequently under Configuring the UTMs later in this section.*

If, after creating a UTM, you submit it to `C:\S4200\kiuser\Tests` or to a personal library, you can subsequently insert it from this library in the same way as an ITM.

This subsection covers the following topics:

- Inserting a completely new UTM
- Inserting a library UTM
- Editing the UTM insertions

CAUTION **If you need to insert multiple instances of an UTM under the same name, be sure that you understand the rules that apply. Review Table 6-4 below.**

Table 6-4
Shared and unique characteristics for same-named Project Plan UTMs

Characteristics that are shared between all instances of a Project Plan UTM having the same name. A change of one instance changes the definition of *all* instances identically.*	**Characteristics that are unique to each instance of a Project Plan UTM having the same name. A change of one instance has no effect on any other instance.**
Everything on the **Definition** tab for the UTM, *except for the parameter settings*. The shared characteristics include the following: • The specified user library • The specified user module • All **Formulator** formulas* • **Output Values**.	• The parameter settings • Test data • Formulas in the **Calc** worksheet of the **Sheet** tab • Plot settings in the **Graph** tab • The Unique IDentifier number (**UID**)

* Changing the user module or the user library causes all **Formulator** formulas to be deleted.

Inserting a completely new UTM

If a UTM of a desired type does not already exist in a test library, you must enter a new UTM into a Project Plan only by name, then connect it to, and configure it for, an existing user module.

When initially building a new Project Plan, it may be convenient to add all new UTMs first without immediately connecting them to user modules; this allows you to focus on Project Plan structure without becoming distracted with configuration details.

NOTE *However, if you intend to insert additional instances of a new UTM under the same name, you must first connect it to a user module and then insert the additional instances from the test library. For information about connecting a UTM to a user library, refer to Configuring the UTMs later in this section. For information about submitting a UTM to a test library, refer to Submitting devices, ITMs, and UTMs to libraries later in this section.*

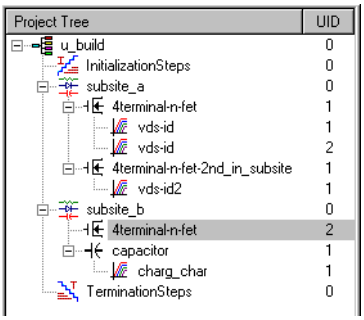
Insert a new, name-only UTM into a Project Plan as follows:

1. In the Project Navigator, select the component below which (or in some cases, above which) you'll insert the UTM. You can insert UTMs at the following places:
   - Below Device Plans, initialization steps, and termination steps.

- Above and below ITMs and other UTMs. The instructions in this subsection reflect adding the UTM below an ITM or UTM.

For illustration purposes, the UTM name will be added to the **u_build** Project Plan below a Device Plan. See Figure 6-72.

Figure 6-72
Selecting the device in which to insert a new UTM name

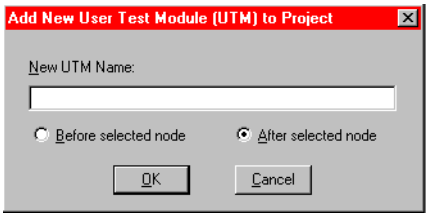


2. Do either of the following:
   - In the **Project** menu, click **New User Test Module.**
   - In the Project Plan toolbar, click the **Add New UTM** button. See below.

Figure 6-73
Add New UTM button

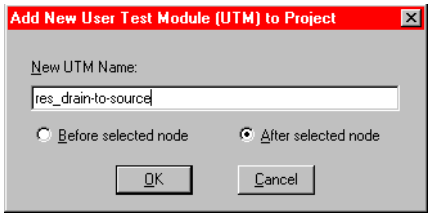
The Add New User Test Module (UTM) to Project dialog box appears. See Figure 6-74.

Figure 6-74
Add New User Test Module (UTM) to Project dialog box



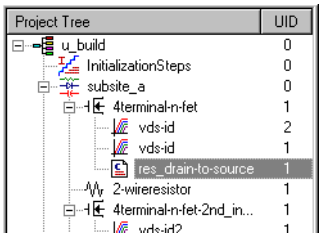
3. In the Add New User Test Module (UTM) to Project dialog box, enter the desired name for the UTM. For the **u_build** Project Plan, the user library to be later connected to the UTM measures the drain-to-source resistance for the FET at saturation. Therefore, the name **res_drain-to-source** is typed in. See Figure 6-75.

Figure 6-75
Entering a new UTM name



4. Click **OK**. The new UTM is inserted into the Project Plan. See Figure 6-76.

Figure 6-76
New UTM entered into the Project Plan



Inserting a library UTM

If a UTM of a desired type exists in a test library, you insert it from the library in exactly the same way as you would insert an ITM from the library, except as follows:

- You can insert a UTM, and only a UTM, below **Initialization Steps** and **Termination Steps** in the Project Navigator.
- If you insert multiple instances of a UTM under the same name, the parameter values are unique for each instance.

With these exceptions in mind, insert library UTMs using the procedures under Inserting ITMs from a test library earlier in this section.

Editing the UTM insertions

For information on editing your UTM insertions, refer to Rearranging and deleting ITMs and UTMs later in this section.

Saving the Project Plan

When you have finished entering the Project Plan, save it by selecting **File** → **Save All** or by clicking the **Save All** toolbar button.

Modifying an existing Project Plan

This subsection helps you to open and edit an existing Project Plan, either to upgrade the Project Plan or to create a completely new one. This subsection also shows you how to delete a Project Plan from within KITE. The topic headings are as follows:

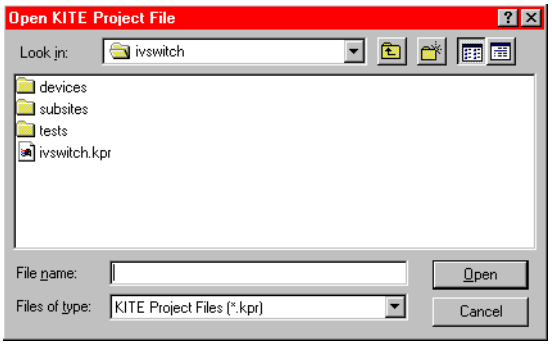
- Opening an existing Project Plan
- Saving a Project Plan under a new name
- Adding and deleting initialization and termination steps
- Adding, rearranging, and deleting Subsite Plans
- Adding, rearranging, and deleting Device Plans
- Rearranging and deleting ITMs and UTMs
- Deleting a Project Plan

Opening an existing Project Plan

To open an existing Project Plan, do the following:

1. In the **File** menu, select **Open Project.** The Open KITE Project File window appears, displaying a file tree for the Project Plan that was last opened in KITE. See Figure 6-77.

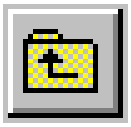
Figure 6-77
Example of Open KITE Project File window as it initially opens



2. In the **File Name** edit box of the Open KITE Project File window, enter the **<ProjectName>.kpr** Project Plan name using one of the following methods:
   - **Method I**: Type **`<ProjectDirectoryPath><ProjectName>.kpr`** directly in the **File Name** edit box.
   - **Method II**: Browse for the file name as follows:

   a. Do one of the following:
      - If the Project Plan is in the default user directory,[4] then click the **next-file-level-up** button, which is illustrated below.

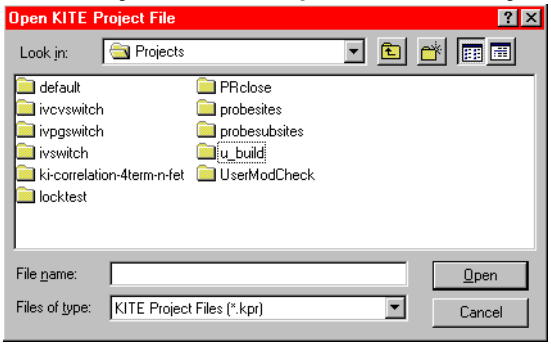
Figure 6-78
Next-file-level-up button



      - If the Project Plan is *not* in the default user directory, in the **Look In** combo box, browse for and insert the correct Project Plan directory (typically `<Path>\Projects`).

   The Open KITE Project File window should now display the folders for all Project Plan files in the default or otherwise specified Project Plan directory. See Figure 6-79.

Figure 6-79
Example display of all Project Plans in specified directory

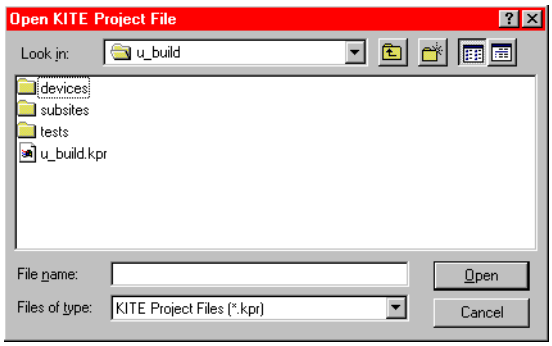


4. For example, the C:\S4200\kiuser\Projects factory-default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Projects.

b. Double-click on the **<ProjectName>** folder (the folder that contains the Project Plan to be opened). The Open KITE Project File window displays the file tree for the Project Plan to be opened.

The **u_build** Project Plan (created previously under Building a completely new Project Plan earlier in this section) is used to illustrate the next procedure (see Saving a Project Plan under a new name). Therefore, the **u_build** folder was selected, resulting in the file tree shown in Figure 6-80.

Figure 6-80
Example of Project Plan file tree in the Open KITE Project File window

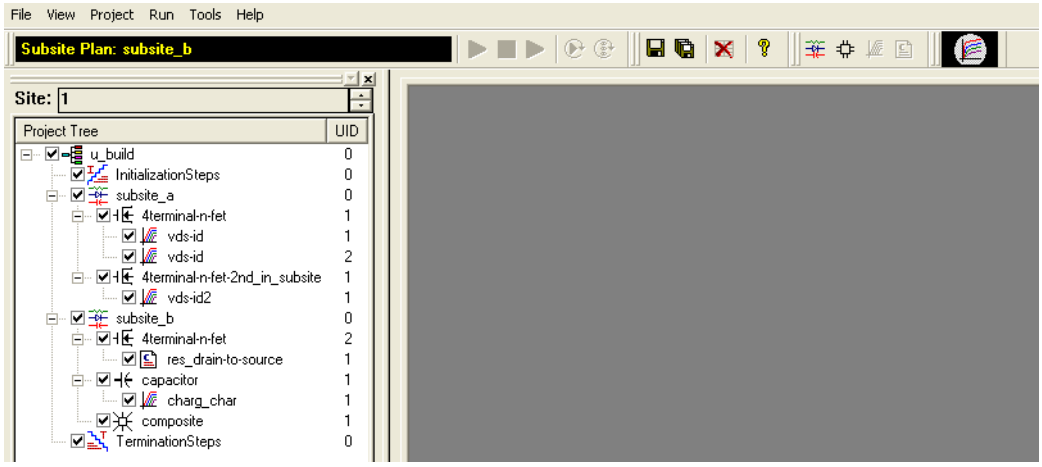


c. Click on the **<ProjectName>.kpr** file name; in our example, **u_build.kpr.** The file name is entered in the **File name** edit box.

3. Click **Open**. The **<ProjectName>** Project Plan opens.

The opened **u_build** Project Plan is shown in Figure 6-81.

Figure 6-81
Opened u_build Project Plan

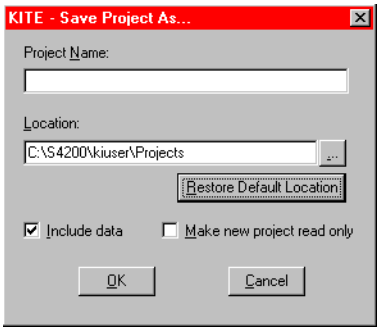


Saving a Project Plan under a new name

To create a completely new Project Plan from an existing source Project Plan, start by saving the source Project Plan under a new name. Do the following:

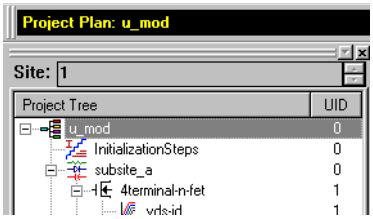
1. Take one or both of the actions below, as required:
   a. If 1) the source Project Plan is already open, 2) you have made unsaved changes to the source Project Plan, and 3) you also wish to save these changes under the source Project Plan file name, be sure to click the **Save All** toolbar button (looks like a stack of diskettes) or select **File → Save All.**
   b. If you need to open a source Project Plan other than the presently open Project Plan, open it as described under Opening an existing Project Plan earlier in this section.
2. In the File menu, select **Save Project As**. The KITE Save Project As dialog box appears. See Figure 6-82.

Figure 6-82
KITE Save Project As dialog box



3. In the KITE Save Project As dialog box, do the following:
   a. In the **Project Name** edit box, type in a new Project Plan name. For illustration purposes, **u_mod** was typed in as the name under which the **u_build** Project Plan was to be saved.
   b. In the **Location** combo box, if the as-displayed destination for the new file is incorrect, select a new location using one of the following:
      - By clicking the **Restore Default Location** button, if you wish to store the new file in the KCON specified default Project Plan directory.
      - By browsing for a new location. Click the key to the right of the combo box to display a file tree, a **Drives** combo box, and a **Network** button (the **Network** button allows you to map a drive on a local-area network [LAN] on which to store the new Project Plan).
   c. In the **Include Data** checkbox, check or uncheck whether you wish to include, in the new Project Plan, the data that was last generated by the source Project Plan.
   d. If you wish to protect the new Project Plan by making it read-only, check the **Make project read only** checkbox.
   e. Click **OK**. The source Project Plan is closed and the new Project Plan is opened in its place. Figure 6-83 shows the new **u_mod** Project Plan that was created from the **u_build** Project Plan using a **Save Project As operation**.

Figure 6-83
New u_mod Project Plan created from the u_build Project Plan using Save Project As



Adding and deleting initialization and termination steps

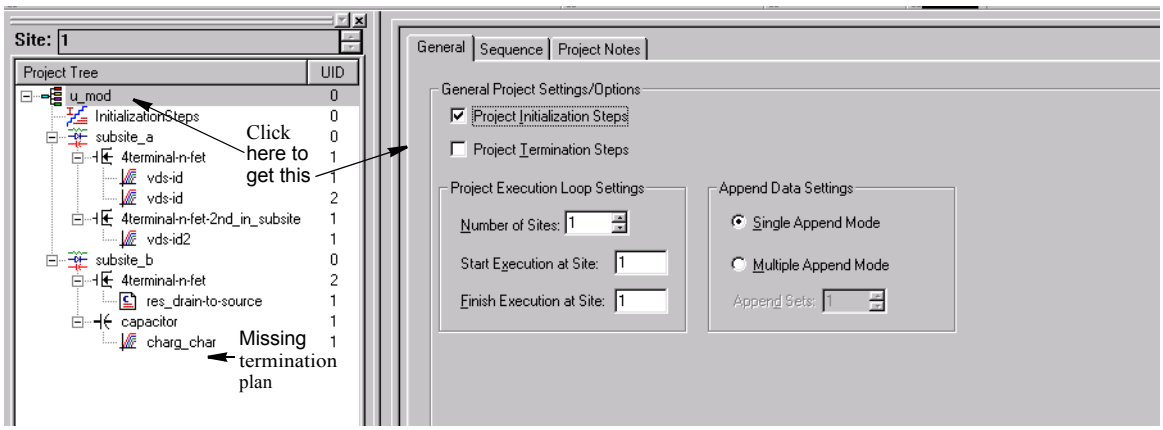
Adding initialization or termination steps

Add initialization or termination steps as follows:

1. Double-click the Project Plan name at the top of the Project Navigator tree. The **General** tab of the Project window opens, displaying **Project Initialization Steps** and **Project Termination Steps** checkboxes.

   For illustration purposes, the termination steps were previously deleted from the **u_mod** Project Plan. Then, the **u_mod** name at the top of the **u_mod** Project Navigator was double-clicked. KITE displayed the image shown in Figure 6-84.

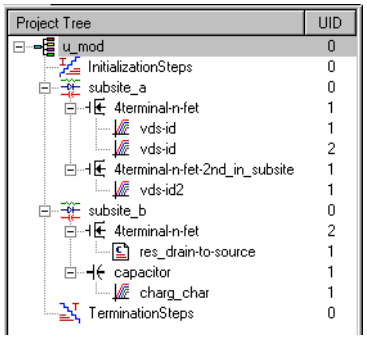
Figure 6-84
Preparing to add initialization or termination steps



2. Check the **Project Initialization Steps** checkbox to add initialization steps. Check the **Project Termination Steps** checkbox to add termination steps.
3. At the lower right corner of the Project window, click the **Apply** button. The initialization steps or termination steps are added to the Project Plan at the correct location(s).

   Figure 6-85 shows the termination steps restored to the **u_mod** Project Plan.

Figure 6-85
Termination steps added



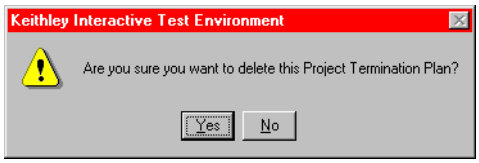
Deleting initialization or termination steps

CAUTION **Deleting initialization or termination steps deletes ALL UTMs in the initialization or termination steps.**

Delete initialization or termination steps as follows:

1. In the Project Navigator, select the intitialization or termination steps.
2. Press the **DELETE** key on the keyboard. A message box appears, asking you to verify the deletion. Figure 6-86 shows the box that appears when deleting termination steps. A similar box appears when deleting initialization steps.

Figure 6-86
Message box that appears when deleting termination steps



3. Click **Yes**. The initialization steps or termination steps, including all UTMs in the plan, are deleted.

Adding, rearranging, and deleting Subsite Plans

Adding a Subsite Plan

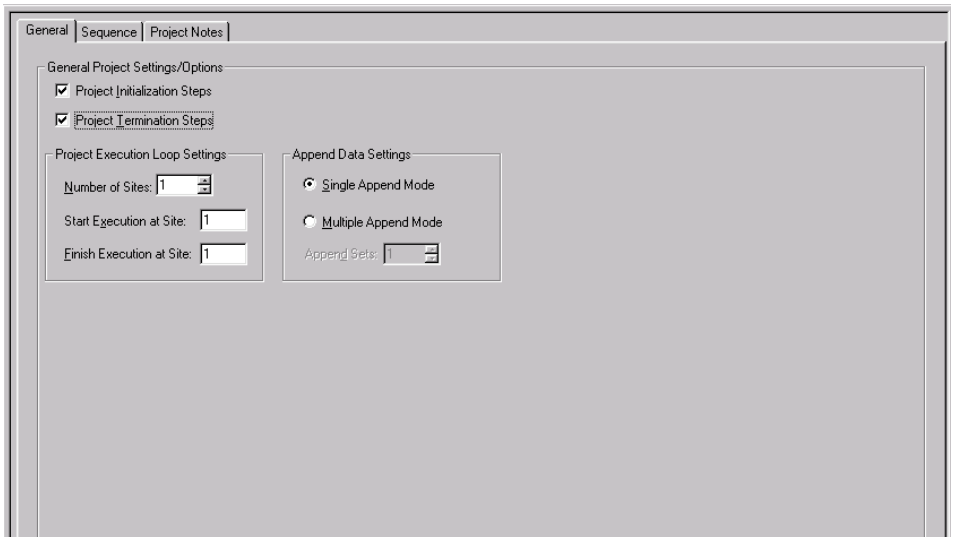
Adding a Subsite Plan to an existing Project Plan is identical to adding a Subsite Plan to a new Project Plan. For instructions, refer to Inserting the Subsite Plans earlier in this section.

Rearranging Subsite Plans

You can rearrange the order of Subsite Plans within a Project Plan, as follows:

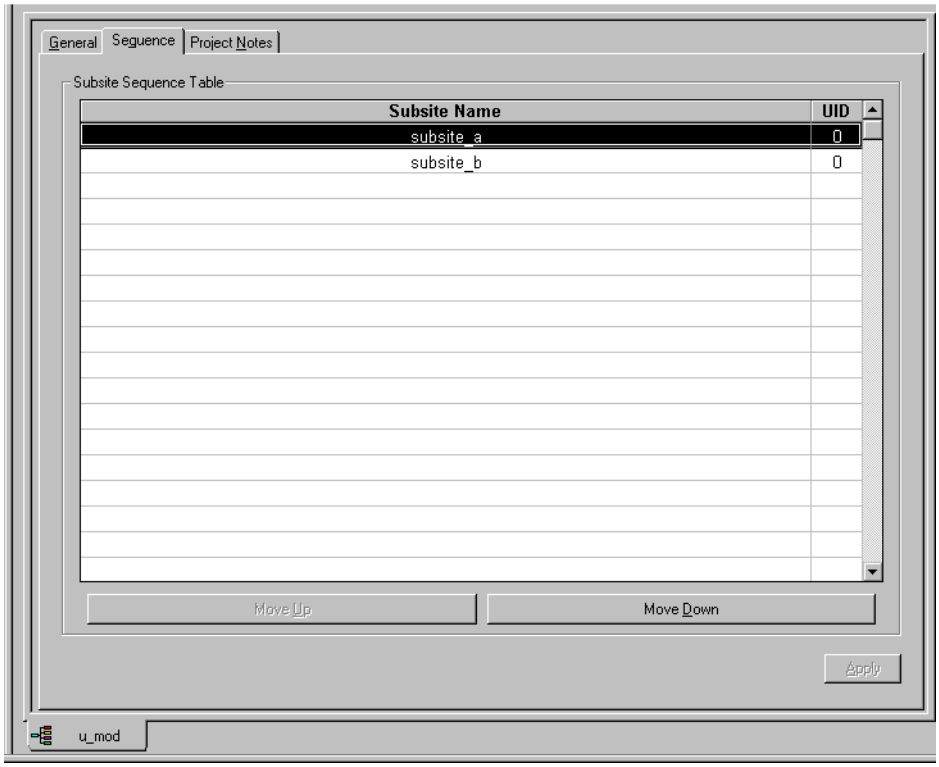
1. In the Project Navigator, double-click the Project Plan (the top-most component in the Project Navigator). The corresponding Project window opens.
   For illustration purposes, the **u_mod** Project Plan was double-clicked, opening the Project window shown in Figure 6-87.

Figure 6-87
Project window



2. Select the **Sequence** tab of the Project window. The **Sequence** tab opens, displaying the **Subsite Sequence Table**. Figure 6-88 shows the **Subsite Sequence Table** for the **u_mod** Project Plan.

Figure 6-88
Subsite Sequence Table for the u_mod Project Plan

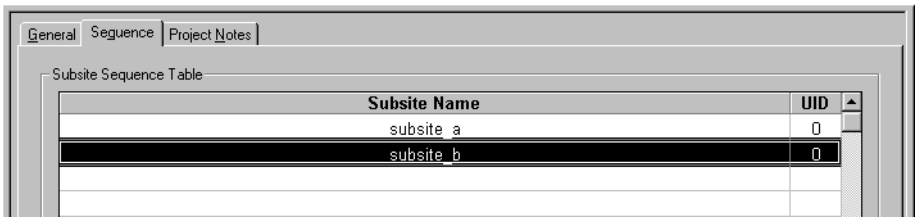


3. Select the Subsite Plan(s) to be moved.

NOTE *To select a sequential group of Subsite Plans, hold down the **SHIFT** key and click on the first and last subsite to be included.*

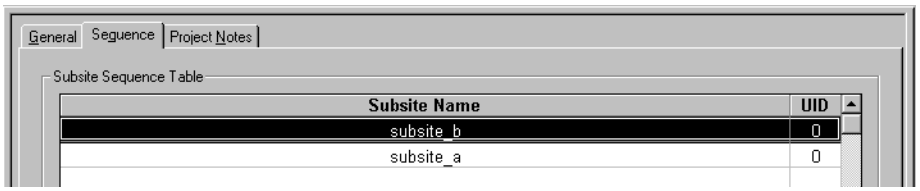
Figure 6-89 shows **subsite_b** selected in the **Subsite Sequence Table** of the **u_mod** Project Plan.

Figure 6-89
Selected subsite_b plan to be moved



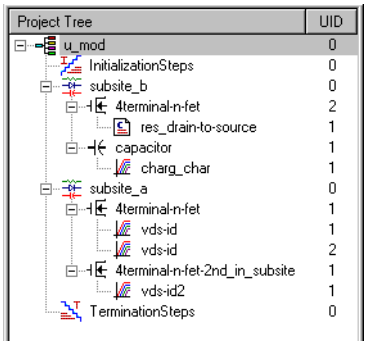
4. Use the Move Up button or the Move Down button to reposition the selected Device Plan. Figure 6-90 shows the Subsite Sequence Table after relocating the **subsite_b** Subsite Plan above the **subsite_a** Subsite Plan, using the **Move Up** button.

Figure 6-90
Relocated subsite_b plan in Subsite Sequence Table



5. Click **Apply** in the lower right corner of the Subsite Plan window. The Device Plan is relocated in the Project Plan. Figure 6-91 shows **subsite_b** relocated in the **u_mod** Project Plan.

Figure 6-91
Relocated subsite_b plan in u_mod Project Plant



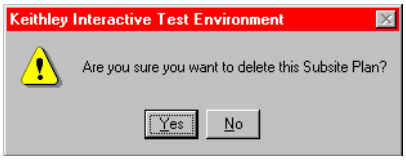
Deleting a Subsite Plan

CAUTION **Deleting a Subsite Plan deletes all Device Plans and tests in the Subsite Plan.**

Delete a Subsite Plan as follows:

1. In the Project Navigator, select the Subsite Plan.
2. Press the **DELETE** key on the keyboard. A message box appears asking you to confirm the deletion. Figure 6-92 shows a typical message box.

Figure 6-92
Message box that appears when deleting a Subsite Plan



3. Click **Yes**. The Subsite Plan, including all Device Plans and tests in the Subsite Plan, are deleted.

Adding, rearranging, and deleting Device Plans

Adding a Device Plan

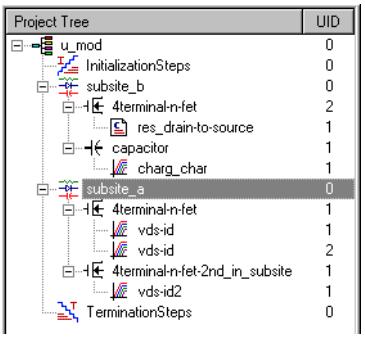
Adding a Device Plan to an existing Project Plan is identical to adding a Device Plan to a new Project Plan. For instructions, refer to Inserting Device Plans earlier in this section.

Rearranging Device Plans

You can rearrange the Device Plans within a Subsite Plan, as follows:

1. In the Project Navigator, locate the Subsite Plan that contains the Device Plan. For illustration purposes, the **4terminal-n-fet-2nd_in_subsite** Device Plan of the **u_mod** Project Plan was chosen for relocation. The Subsite Plan that contained this Device Plan was **subsite_a**. See Figure 6-93.

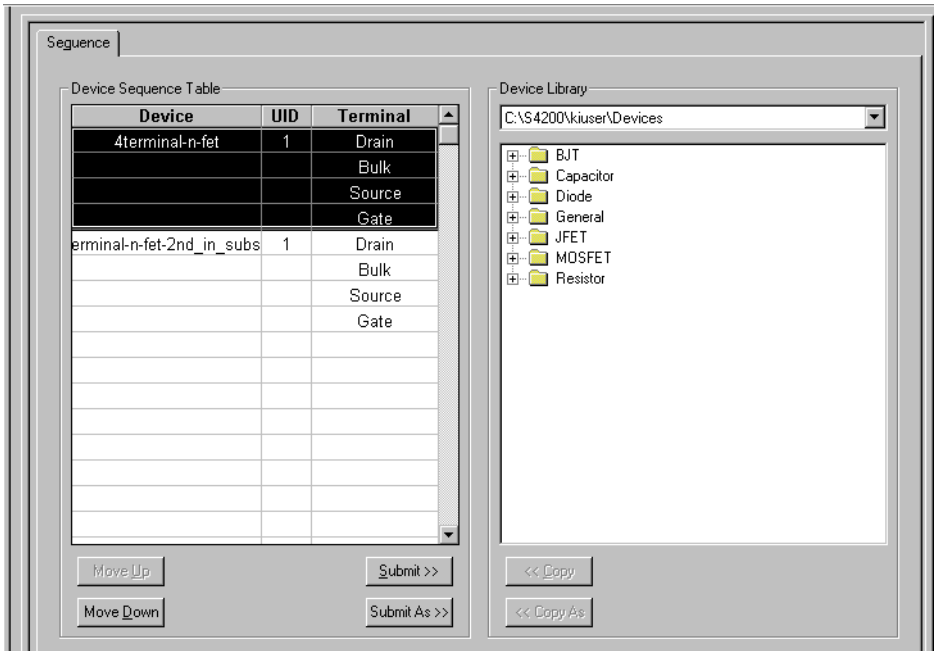
Figure 6-93
Subsite Plan containing a Device Plan to be moved



2. Double-click the Subsite Plan that contains the Device Plan to be relocated. The corresponding Subsite Plan window opens.

   For illustration purposes, the **subsite_a** Device Plan in the **u_mod** Project Plan was double-clicked, opening the Subsite Plan window shown in Figure 6-94.

Figure 6-94
Subsite Plan window opened for 4terminal-n-fet-2nd_in_subsite Device Plan to be relocated

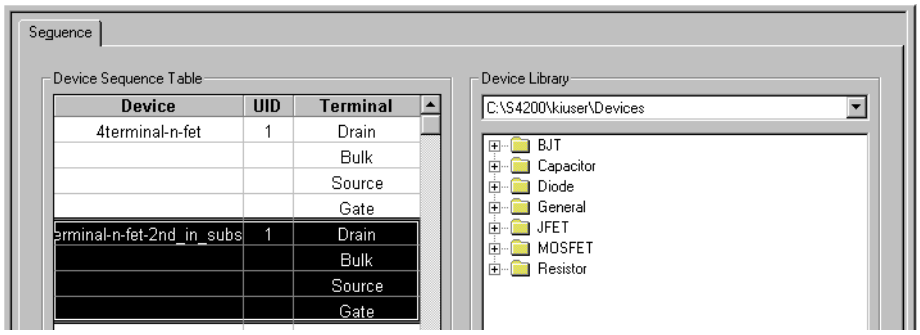


3. In the **Device Sequence Table** of the Subsite Plan window, select the Device Plan(s) to be moved.

NOTE *To select a sequential group of Device Plans, hold down the **SHIFT** key and click on the first and last Device Plan to be included.*

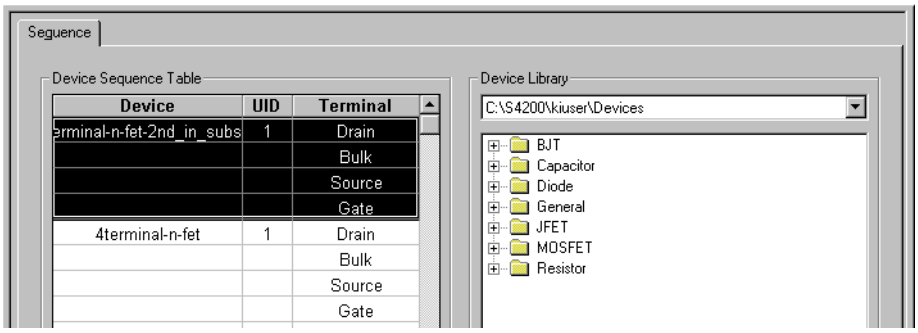
Figure 6-95 shows the **4terminal-n-fet-2nd_in _subsite** Device Plan selected in the **u_mod** Project Plan.

Figure 6-95
Device Sequence Table selection of Device Plan to be moved



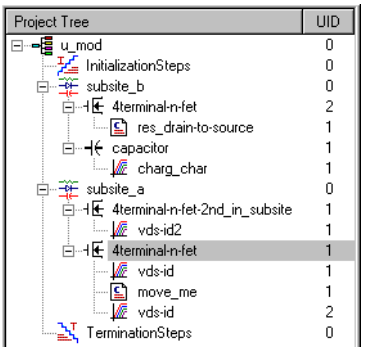
4. Use the **Move Up** button or the **Move Down** button to reposition the selected Device Plan. Figure 6-96 shows the result of relocating the **4terminal-n-fet_2nd_in _subsite** Device Plan above the **4terminal-n-fet** Device Plan, using the **Move Up** button.

Figure 6-96
Relocated 4terminal-n-fet-2nd_in _subsite Device Plan in Device Sequence Table



5. Click **Apply** in the lower right corner of the Subsite Plan window. The Device Plan is relocated in the Project Plan. See Figure 6-97.

Figure 6-97
Relocated 4terminal-n-fet-2nd_in _subsite Device Plan in the u_mod Project Plan



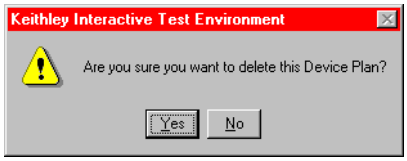
Deleting a Device Plan

CAUTION **Deleting a Device Plan deletes ALL tests in the Device Plan.**

Delete a Device Plan as follows:

1. In the Project Navigator, select the Device Plan.
2. Press the **DELETE** key on the keyboard. A message box appears asking you to confirm the deletion. Figure 6-98 shows a typical message box.

Figure 6-98
Message box that appears when deleting a Device Plan



3. Click **Yes**. The Device Plan, including all tests in the Device Plan, are deleted.

Rearranging and deleting ITMs and UTMs

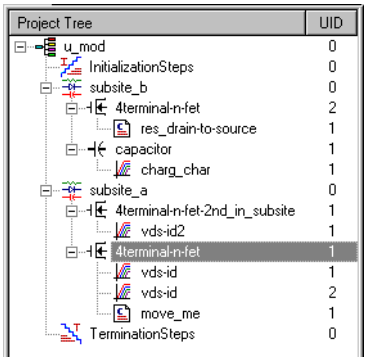
Rearranging ITMs and UTMs

You can rearrange the order of ITMs and UTMs within a Device Plan, as follows:

1. In the Project Navigator, locate the Device Plan that contains the ITM(s) or UTM(s) to be relocated (hereafter, simply referred to as test(s) if appropriate).

   For illustration purposes, the **move_me** UTM was added to the **u_mod** Project Plan and was chosen for relocation. The **4terminal-n-fet** Device Plan in **subsite_a** contains the **move_me** UTM. See Figure 6-99.

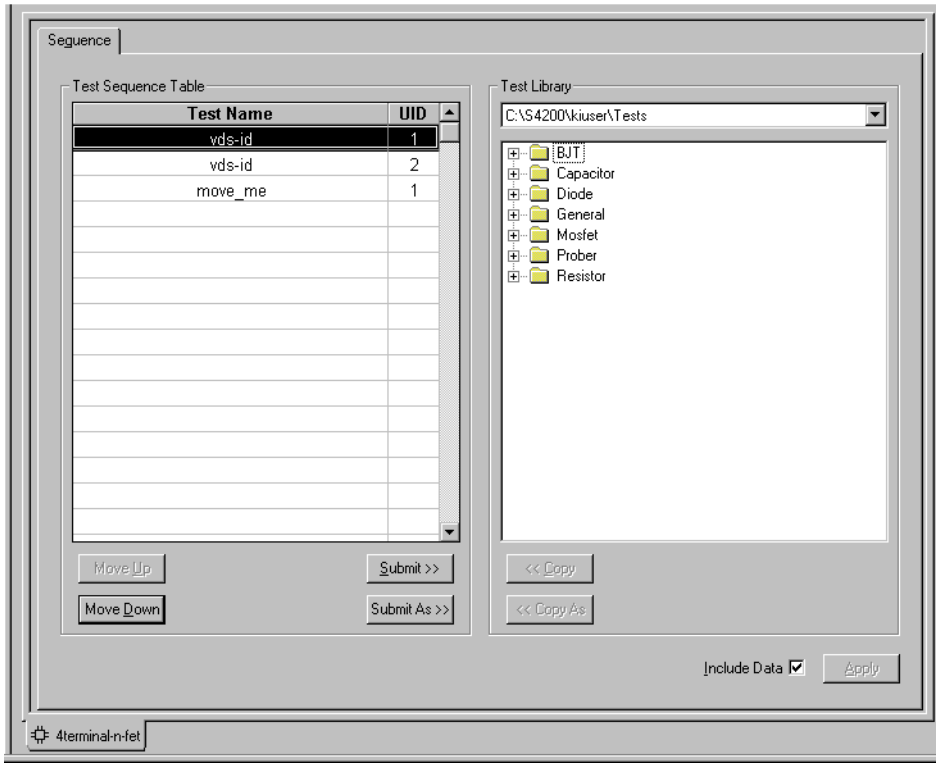
Figure 6-99
Device Plan containing a UTM to be moved



2. Double-click the Device Plan that contains the test to be relocated. The corresponding Device Plan window opens.

   For illustration purposes, the **4terminal-n-fet** Device Plan was double-clicked, opening the Device Plan window shown in Figure 6-100.

Figure 6-100
Device Plan window opened for the move_me UTM to be relocated

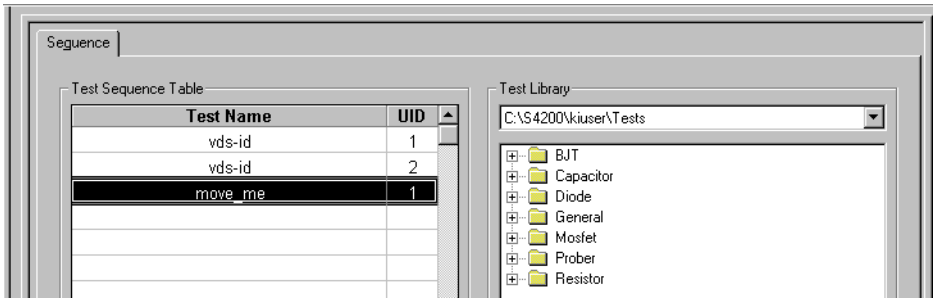


3. In the **Test Sequence Table** of the Device Plan window, select the test(s) to be moved.

NOTE *To select a sequential group of tests, which can be a mixture of ITMs and UTMs, hold down the **SHIFT** key and click on the first and last test to be included.*

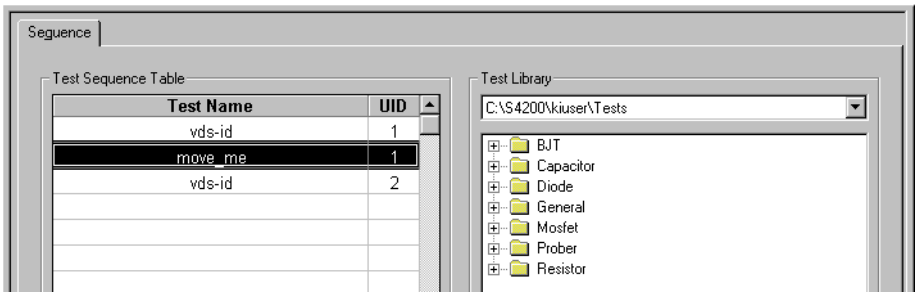
Figure 6-101 shows the **move_me** UTM selected.

Figure 6-101
Test Sequence Table selection of move_me UTM to be moved



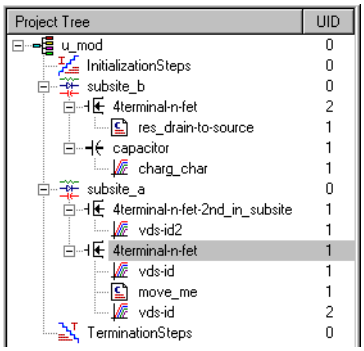
4. Use the Move Up button or the Move Down button to reposition the ITM or UTM.
   Figure 6-102 shows the result of relocating the **move_me** UTM to between the two **"vds-id"** ITMs, using the **Move Up** button.

Figure 6-102
Relocated move_me UTM in Test Sequence Table



5. Click **Apply** in the lower right corner of the Device Plan window. The ITM or UTM is relocated in the Project Plan. See Figure 6-103.

Figure 6-103
Relocated move_me UTM in Project Plan

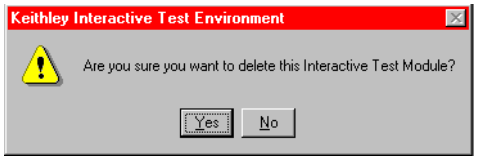


Deleting an ITM or UTM

Delete an ITM or UTM as follows:

1. In the Project Navigator, select the ITM or UTM.
2. Press the **DELETE** key on the keyboard (alternatively, right-click on the ITM or UTM and, in the pop-up menu that appears, select **Delete**). A message box appears asking you to confirm the deletion. Figure 6-104 shows a typical message box that appears when deleting an ITM. A similar box appears when deleting a UTM.

Figure 6-104
Message box that appears when deleting an ITM



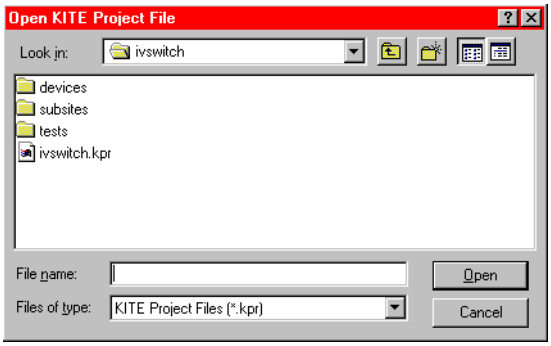
3. Click **Yes**. The ITM or UTM is deleted.

Deleting a Project Plan

CAUTION	**It is possible to delete a Project Plan from within KITE, without opening the Windows Explorer. However, before deleting a Project Plan, ensure that you and others will not need it in the future.**

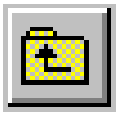
1. In the **File** menu, select **Open Project.** The Open KITE Project File window appears, displaying a file tree for the Project Plan that was last opened in KITE. See Figure 6-105.

Figure 6-105
Example of Open KITE Project File window as it initially opens



2. In the large central area of the Open KITE Project File window, display the **<ProjectName>** folder of the Project Plan to be deleted, as follows:
   - If the Project Plan to be deleted is located in the default user directory,[5] click the next-file-level-up button (illustrated below).

Figure 6-106
Next-file-level-up button



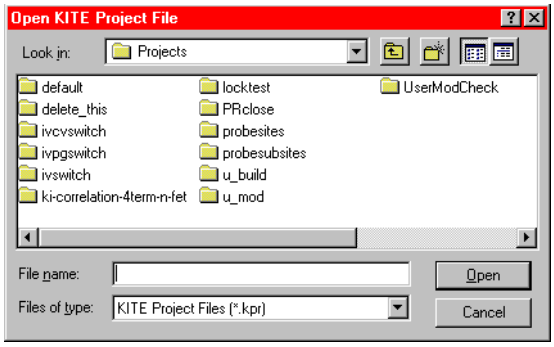
   - If the **Project Plan** to be deleted is not in the default user directory, in the **Look In** combo box browse for and insert the correct Project Plan directory (typically **<Path>\Projects**).

   The Open KITE Project File window should now display the folders for all Project Plan files in the default or other specified Project Plan directory.

   In Figure 6-107, note that the Open KITE Project File window displays a folder for the **delete_this** Project Plan.

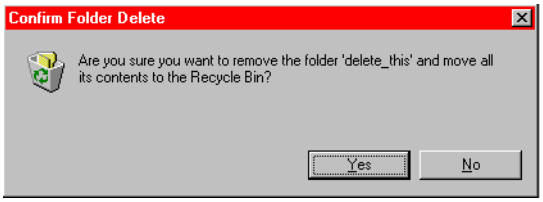
5. For example, the C:\S4200\kiuser\Projects factory-default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Projects.

Figure 6-107
Example of Open KITE Project File window, displaying all Project Plans in the specified directory



3. Select the **<ProjectName>** folder of the Project Plan to be deleted. For illustration purposes, the **delete_this** folder that was shown in Figure 6-107 was selected.
4. Press the **DELETE** key. A message box appears, naming the Project Plan and asking you to confirm the deletion. Figure 6-108 shows a typical message box.

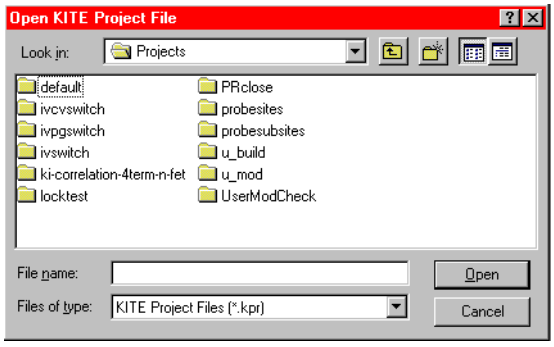
Figure 6-108
Message box that appears when deleting a Project Plan



NOTE *If you hold down the SHIFT key while pressing the **DELETE** key, the Project Plan is removed without placing it in the recycle bin.*

5. Click **Yes**. The Project Plan is deleted, as can be seen in the Open KITE Project File window. See Figure 6-109.

Figure 6-109
Open KITE Project File window, reflecting deletion of the delete_this Project Plan



6. In the Open KITE Project File window, click **Cancel**. The Open KITE Project File window closes.

Configuring the Project Plan ITMs

After inserting ITMs and UTMs into your Project Plan, they must be configured to meet your test requirements. This subsection describes use of the powerful and flexible features of KITE to configure your Project Plan ITMs. UTM configuration is discussed subsequently under Configuring the UTMs later in this section.

CAUTION **If your Project Plan contains multiple same named instances of the ITM to be configured/reconfigured, ensure that you understand the shared and unique characteristics of same named ITMs before configuring/ reconfiguring an ITM. Refer to Table 6-5 below.**

Table 6-5
Shared and unique characteristics for same-named Project Plan ITMs

Characteristics that are shared between all instances of a Project Plan ITM having the same name. A change of one instance changes the definition of *all* instances identically.*	**Characteristics that are unique to each instance of a Project Plan ITM having the same name. A change of one instance has no effect on any other instance.**
Everything on the **Definition** tab for the ITM, including the following: • The instrument selections for each instrument object. • The instrument settings for each selected instrument (the configurations implemented using the Forcing Function/Measure Options windows for all device terminals). • All **Formulator** formulas.* • All **Timing** settings. • **Exit Conditions**. • **Output Values**. • The **Speed** selection. • The **Mode** selection.	• Test data • Formulas in the **Calc** worksheet of the **Sheet** tab • Plot settings in the **Graph** tab • The Unique IDentifier number (**UID**)

* Some changes to an ITM result in the deletion of all **Formulator** formulas.

Configuration of ITMs is discussed under the following topics:

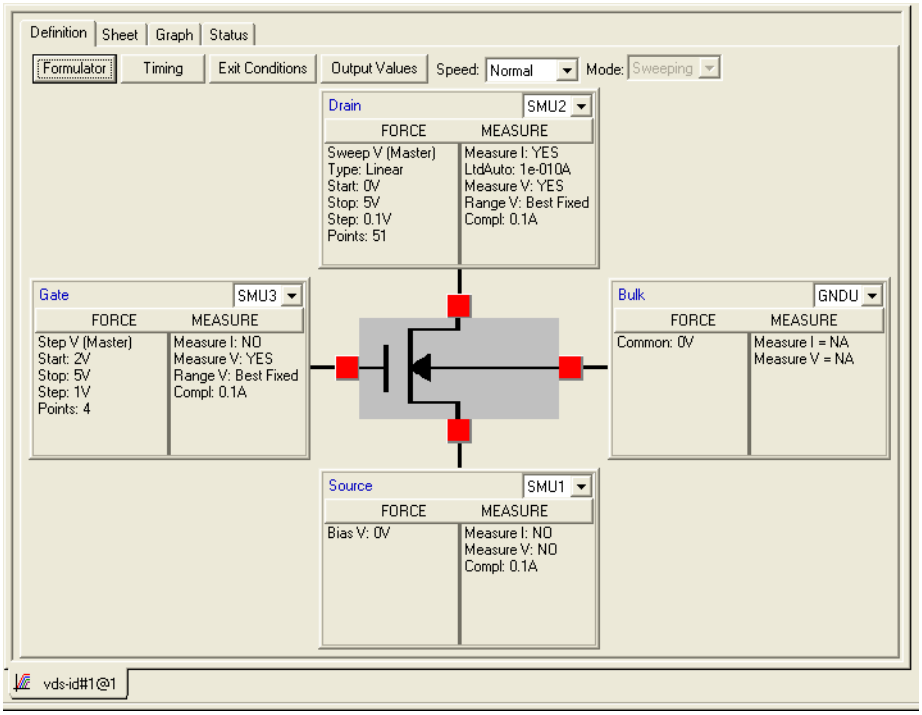
- Opening an ITM window
- Becoming acquainted with the ITM Definition tab
- Matching Definition tab terminal connections to physical connections
- Selecting the ITM test mode
- Assigning/reassigning forcing functions to the device terminals
- Configuring SMU Forcing Functions/Measure Options window
- Configuring the Speed and Timing settings in the ITM Definition tab
- Configuring Formulator calculations
- Saving the ITM configuration
- ITM compliance exit conditions
- ITM Output Values

Opening an ITM window

Each ITM is associated with a specific ITM window. An ITM window is the user interface to every aspect of an ITM: its definition/configuration, its numerical and graphical test data, and its data analysis.

To open an ITM window, in the Project Navigator double-click on the ITM that you wish to configure. The **Definition** tab of the ITM window opens by default. Also, the labels of the **Sheet** tab, **Graph** tab, and **Status** tab are displayed in the ITM window for ready access to these tabs. Figure 6-110 shows an example ITM window.

Figure 6-110
ITM window displaying its Definition tab



NOTE *An ITM window for a chosen ITM may already be open but hidden behind another ITM or UTM window. If so, double-click on the ITM in the Project Navigator. The ITM window will be brought into the foreground (If Workbook Mode has been selected in the Options window [**Tools > Options**], you can bring an ITM window into the foreground by clicking on its displayed Workspace window tab).*

The four tabs of the ITM window are used as follows:

- The **Definition** tab is the primary interface for configuring an ITM. A **Definition** tab allows you to configure an interactive test and display the current configuration. The included **Formulator** analysis tool of the **Definition** tab allows you to perform calculations on test results and include the calculation results in **Sheet** and **Graph** tabs (see the following) (in most cases, in real time as the ITM executes).
- The **Sheet** tab displays the test results in its **Data** worksheet, in spreadsheet format and in real time as the test executes. An independent spreadsheet in the **Sheet** tab, the **Calc** worksheet, allows the user to perform custom, test-specific data analysis. A third worksheet, the **Settings** worksheet, displays the same settings information as in the **Definition** tab, but

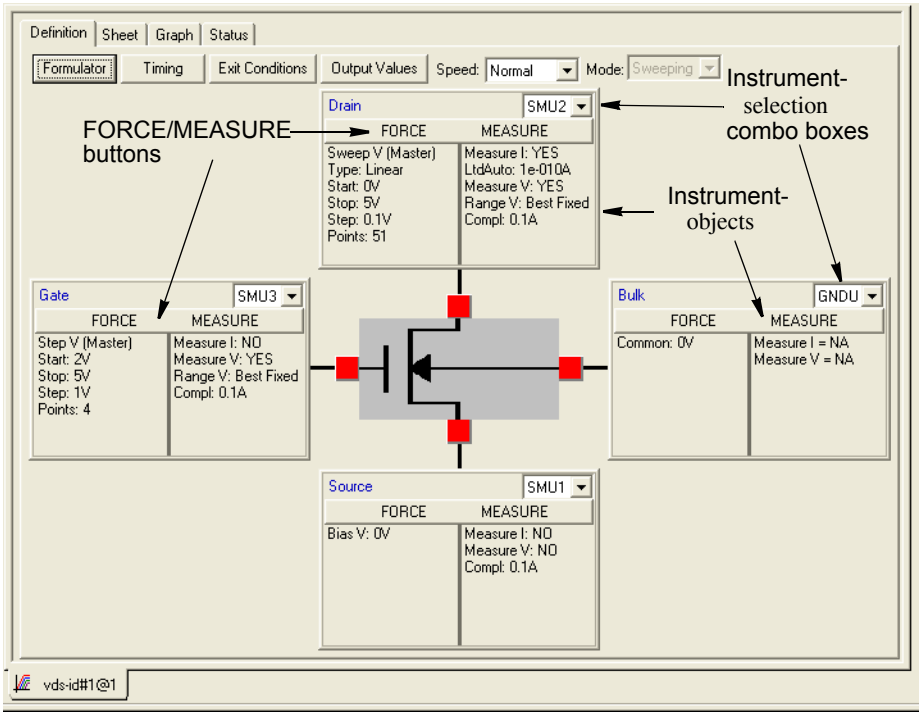
in spreadsheet format. Optional **Append** executions generate a fourth type of worksheets: **Append1**, **Append2** ... and so on.

- Cells in the **Calc** worksheet may be hot-linked to cells in **Data**, **Settings**, and **Append** worksheets. The **Data**, **Settings**, and **Append** worksheets are read-only. However, you may modify the contents of the **Calc** worksheet.
- The **Graph** tab allows the user to create and export graphs of the test and test analysis results, which in most cases may be displayed in real time as the ITM executes. The **Graph** tab provides for flexible plot-data selection, formatting, annotation, and numerical coordinate display (using precision cursors).
- The **Status** tab monitors the configuration status of the test and provides resolution suggestions if there are configuration problems.

Becoming acquainted with the ITM Definition tab

Figure 6-111 shows an example **Definition** tab for a library FET test. Added callouts contextually define terms that are used subsequently in the manual.

Figure 6-111
ITM Definition tab example



An ITM **Definition** tab does the following:

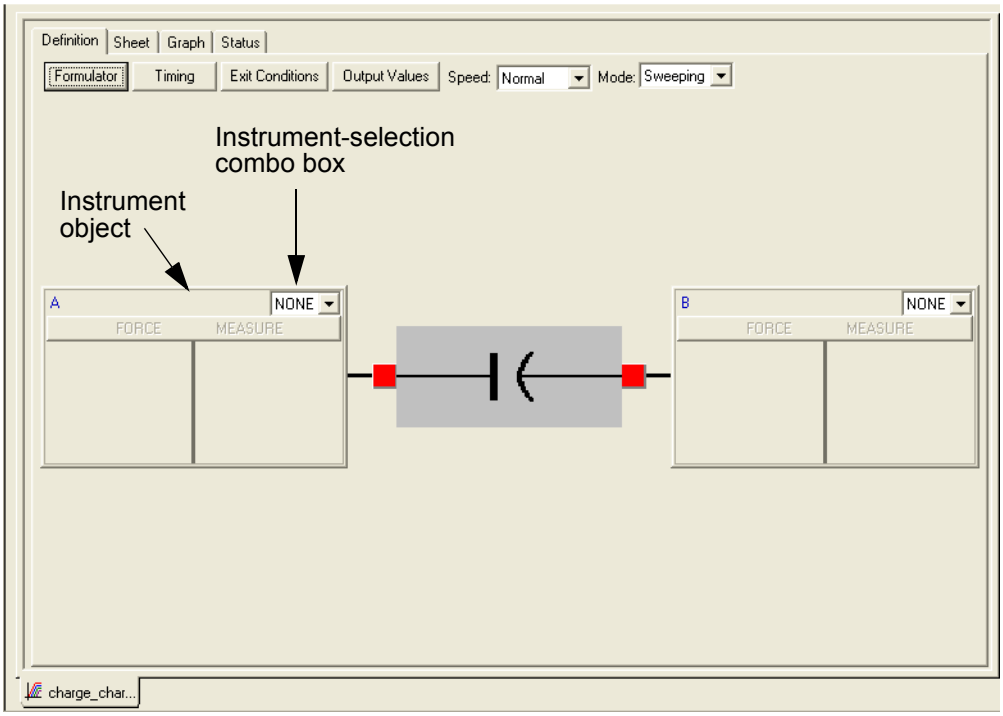
- Displays the test device schematically.
- Displays an instrument object next to each terminal of the device under test. The instrument object represents the SMU (or ground or open circuit) to which the device terminal is connected and provides the following:
  - Identifies the terminal (for example, as gate, drain, source, collector, anode, and so on).
  - Through its instrument-selection combo box allows each terminal to be assigned to match the SMU, GNDU or open circuit that is physically connected to the terminal during the test.
  - Displays the present forcing-function and measuring-options configurations for the terminal.
  - For each terminal that is connected to an SMU, allows assignment and configuration/reconfiguration of the forcing function and measuring options. A single-click of the **FORCE MEASURE** button of the associated instrument object displays a Forcing Functions/Measure Options window for the terminal.
- Provides access to the **Formulator**, which allows simple in-test and complex post-test data computations.
- Allows you to set **Exit Conditions**.
- Allows you to select **Output Values** to be exported to the Subsite Data sheet.
- Allows setting of preconfigured **Speed** or custom **Timing** parameters for the ITM.
- Displays the current test **Mode, Sweeping** or **Sampling,** for all configurations, and allows preselection of the **Mode** prior to configuring a completely new ITM (preselection of the mode simplifies the configuration process).

The ITM window displaying the **Definition** tab shown in Figure 6-111 was opened by double-clicking a **"vds-id"** ITM in the Project Navigator for the **u_build** Project Plan (for more information about the **u_build** Project Plan, refer to a preceding subsection, Building a completely new Project Plan earlier in this section). The **"vds-id"** ITM is an existing, library ITM, and comes preconfigured with default settings.

If you double-click a completely new ITM in the Project Navigator, the ITM window opens and displays a blank **Definition** tab. Based on the device that owns the ITM, the appropriate number of instruments are displayed (one for each device terminal).

Figure 6-112 shows the blank **Definition** tab for the **charg_char** ITM. The **charg_char** ITM is an undefined capacitor test that was inserted into the **u_build** illustration Project Plan during the course of Building a completely new Project Plan earlier in this section.

Figure 6-112
Blank Definition tab for a completely new ITM



All configuration information remains to be supplied for the **charge_char** ITM of Figure 6-112.

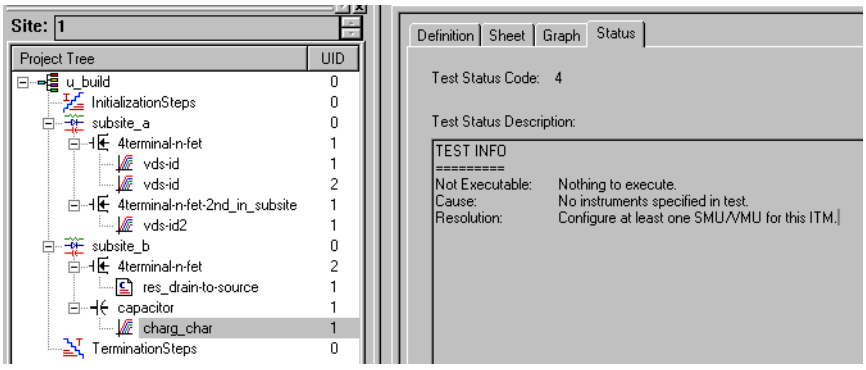
Remaining subsections under Configuring the Project Plan ITMs earlier in this section further explain the features of an ITM **Definition** tab and how they are used to configure an ITM.

ITM Status tab

Before configuring an ITM you can check the present configuration status of the ITM through the ITM **Status** tab. The **Status** tab reports the ITM's readiness for use and typically recommends additional preparations if necessary.

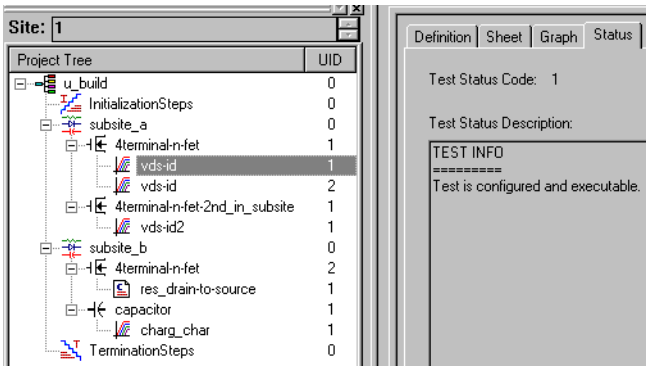
For example, a check of the **Status** tab for the **charge_char** ITM, per Figure 6-112 above, shows its improper configuration status and suggests required actions. See Figure 6-113.

Figure 6-113
Status tab report for the unconfigured charge_char ITM



By contrast, the **Status** tab indicates that the first **"vds-id"** ITM of the **u_build** Project Plan is ready to execute. See Figure 6-114.

Figure 6-114
Status tab report for the configured "vds-id" ITM



Matching Definition tab terminal connections to physical connections

CAUTION **The software connections must accurately reflect the physical hardware connections at the time the ITM is executed. Incorrect terminal configurations result in anomalous test results at best or device damage at worst.**

Match the **Definition** tab terminal connections to the physical connections as follows:

1. Inventory the physical connection of each test device terminal shown in the **Definition** tab, whether it is unconnected (open circuit), connected to the Ground Unit (GNDU), or connected to a particular SMU.
2. Check the instrument selection combo box for each device to see if the **Definition** tab terminal connection is already properly matched to a physical connection.
3. For each device terminal in the **Definition** tab that is improperly matched, or not yet matched, to a physical connection, assign/reassign the terminal connection as follows:
   a. Using the arrow key next to the instrument selection combo box, display a list of possible connections.
   b. Select the connection in the instrument selection combo box that matches the physical connection of the device terminal.

Figure 6-115 illustrates assignment of a terminal connection.

Figure 6-115
Assigning a terminal connection in the Definition tab

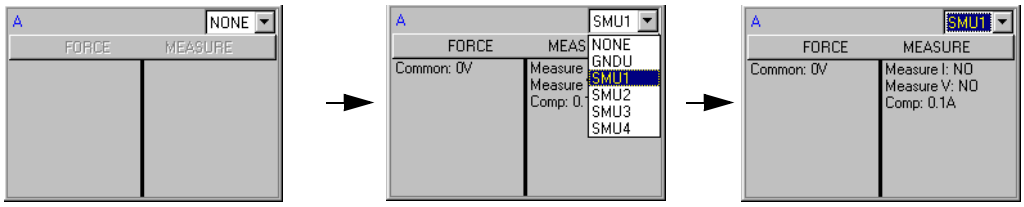
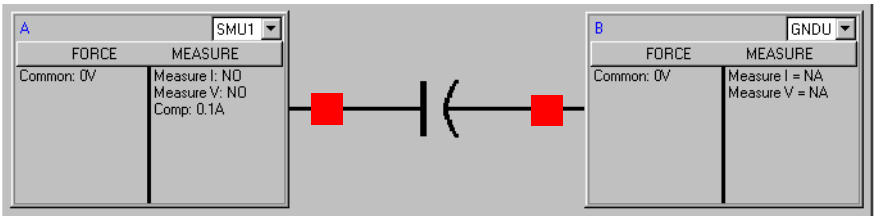


Figure 6-116 shows instrument objects of the previously blank **Definition** tab for the **charg_char** ITM. The terminal settings (Figure 6-112) are now matched to physical connections: SMU1 and the Ground Unit (GNDU).

Figure 6-116
Connected terminal settings example



In Figure 6-116, note that instrument object connected to SMU1 is now assigned, by default, to an unconfigured **Common** forcing function (to be described later).

To configure or reconfigure each SMU, continue with the next five subsections, Selecting the ITM test mode, Assigning/reassigning forcing functions to the device terminals, Configuring SMU Forcing Functions/Measure Options window, and Configuring Formulator calculations later in this section.

Selecting the ITM test mode

The **Sweeping** test mode applies to any ITM in which one or more forced voltages/currents vary with time. The **Sampling** test mode applies to any ITM in which all forced voltages or currents are static, with measurements typically being made at timed intervals. For example, sampling mode would be used to record a few static measurements or to time profile the charging voltage of a capacitor while forcing a constant current.

Understanding the Mode combo box

The **Mode** combo box (Figure 6-111) is used to observe or change the test mode as follows:

- For a completely new ITM, the **Mode** combo box allows you to select the **Sweeping** or **Sampling** test mode. Selecting the appropriate test mode simplifies configuration options and helps to avoid errors, as follows.
  - Only the static forcing functions are configurable in the **Sampling** mode.
  - Only mode-appropriate timing options are configurable (refer to Timing window later in this section).
- For an existing, library ITM that is in the **Sampling** mode, the **Mode** combo box allows you to change to the **Sweeping** mode if you wish to change some of the static forcing functions to dynamic forcing functions.
- For an existing library ITM that is in the **Sweeping** mode, the **Mode** combo box allows you only to *observe* that it is in the **Sweeping** mode. You cannot change the **Sweeping** mode to **Sampling** mode, unless you first change all of the dynamic forcing functions of the ITM to static forcing functions. This lockout helps to avoid errors by allowing only mode-appropriate timing options (see Timing window later in this section).

Selecting Sweeping or Sampling Mode

Select the test mode prior to configuring a completely new ITM, or prior to reconfiguring an existing ITM that is currently in the **Sampling** mode. Select the test mode by first clicking the scroll arrow at the right of the Mode combo box and then clicking on the desired mode.

NOTE *If you select the sampling mode, you must ultimately specify the sampling interval and number of samples in the ITM timing window (otherwise, KITE uses default*

parameters). This is discussed later under Understanding and configuring the Sweeping Mode area of the Timing window later in this section.

Assigning/reassigning forcing functions to the device terminals

A forcing function defines how an SMU applies a current or voltage to a device terminal (including possibly maintaining a zero-voltage/current state). You must initially assign at least one forcing function to a completely new ITM. You can reassign a forcing function(s) to an existing, library ITM, thereby effectively converting it to a new ITM. This subsection helps you to assign forcing functions.

Reviewing the available forcing functions

KITE provides four forcing functions in the Sampling test mode and six additional forcing functions in the Sweeping test mode (for more information about test modes, refer to the last subsection). These functions are summarized in Table 6-6. The forcing functions that can provide pulse output are also identified in Table 6-6 (see Pulse Mode later in this section).

Table 6-6
Forcing function summary

Category	Name	Description	Availability		
			Sweeping	Sampling	Pulse Mode
Static	Open	Maintains a zero-current state at the terminal, subject to the maximum voltage compliance of the connected SMU.	•	•	
	Common	Maintains a zero-voltage state at the terminal, subject to the maximum current compliance of the connected SMU.	•	•	
	Current Bias	Maintains a selected constant-current state at the terminal, subject to the user-specified voltage compliance for the connected SMU.	•	•	•
	Voltage Bias	Maintains a selected constant-voltage state at the terminal, subject to a user-specified current compliance of the connected SMU.	•	•	•
Sweep	Current Sweep	Increments a series of current values at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. Generates parametric curve data that is recorded in the **Sheet** tab **Data** worksheet for the ITM and can be plotted in the ITM **Graph** tab. A dual current sweep can also be performed by an SMU (see Understanding a Dual Sweep later in this section).	•		•
	Voltage Sweep	Increments a series of voltage values at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. Generates parametric curve data that is recorded in the ITM **Sheet** tab **Data** worksheet and can be plotted in the ITM **Graph** tab. A dual voltage sweep can also be performed by an SMU (see Understanding a Dual Sweep later in this section).	•		•
List sweep	Current List Sweep	Steps through a list of user-specified current values, at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. Generates parametric data that is recorded in the ITM **Sheet** tab **Data** worksheet and can be plotted in the ITM **Graph** tab, if appropriate.	•		•
	Voltage List Sweep	Steps through a list of user-specified voltage values, at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. Generates parametric data that is recorded in the ITM **Sheet** tab **Data** worksheet and can be plotted in the ITM **Graph** tab, if appropriate.	•		•
Step	Current Step	Increments a current to two or more levels, each of which is held constant during the progress of a Current Sweep, a Voltage Sweep, a Current List Sweep, or a Voltage List Sweep at another terminal. For each Current Step level, parametric curve data is recorded in the ITM **Sheet** tab **Data** worksheet. The combined data can be plotted in the ITM **Graph** tab, resulting in a series (family) of curves.	•		
	Voltage Step	Increments a voltage to two or more levels, each of which is held constant during the progress of a Current Sweep, a Voltage Sweep, a Current List Sweep, or a Voltage List Sweep at another terminal. For each Voltage Step level, parametric curve data is recorded in the ITM **Sheet** tab **Data** worksheet. The combined data can be plotted in the ITM **Graph** tab, resulting in a series (family) of curves.	•		

For detailed information about each forcing function, refer to The ForcingFunctionName function parameters area later in this section.

Assigning forcing functions for a completely new ITM

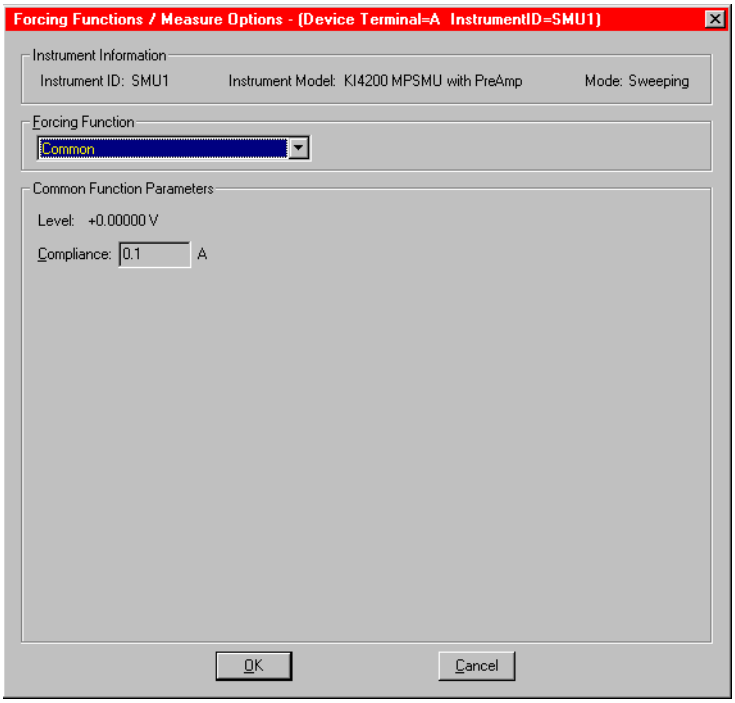
As illustrated in Figure 6-116, all SMU-connected terminals for a completely new ITM are assigned by default to a **Common** forcing function. For the ITM to be useful, at least one of the device terminals must be reassigned to a forcing function that applies a voltage or current. However, to access the new forcing function, you must first open the default, **Common** Forcing Functions/ Measure Options window for the terminal.

NOTE *A Forcing Functions/Measure Options window is the configuration interface for an SMU. The window is used to configure the selected forcing function and measurements implemented by the SMU. Forcing Functions/Measure Options windows are subsequently discussed in detail under Configuring SMU Forcing Functions/Measure Options window later in this section.*

Opening a Common default Forcing Function/Measure Options window for a terminal

To open the **Common** default Forcing Function/Measure Options window for a terminal, click the **FORCE MEASURE** button of the associated instrument object. The **Common** window appears. See Figure 6-117.

Figure 6-117
Common Forcing Functions/Measure Options window



Replacing the default forcing function with a new forcing function

To assign a replacement forcing function to a device terminal, do the following:

1. On the Forcing Function/Measure Options window, click the arrow key next to the **Forcing Functions** combo box. A list of the available forcing functions appears. Figure 6-118 shows the available selections for the **Sampling** test mode. Figure 6-119 shows the available selections for the **Sweeping** test mode.

Figure 6-118
Forcing functions available for the Sampling test mode

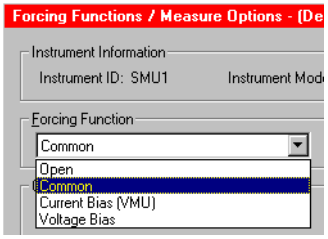
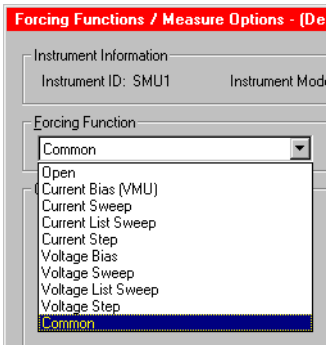
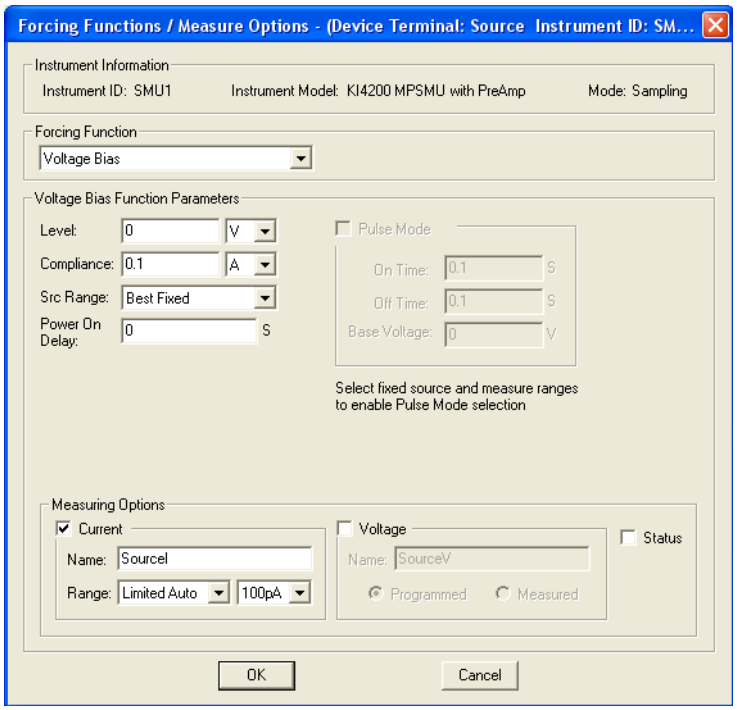


Figure 6-119
Forcing functions available for the Sweeping test mode



2. In the displayed list, click on the desired forcing function. A new Forcing Function/Measure Options window appears. Figure 6-120 shows a window that appeared when the **Voltage Bias** forcing function was clicked to replace the **Common** forcing function (in this case, for the **charg_char** ITM shown previously in Figure 6-116).

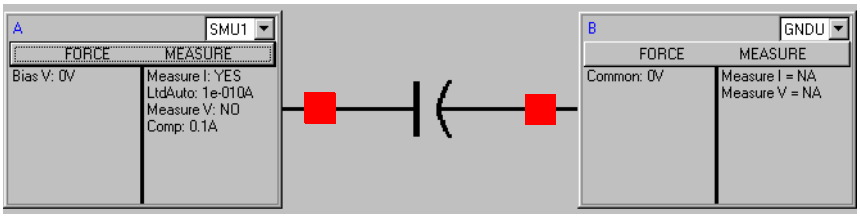
Figure 6-120
Example of a reassigned forcing function



3. Click **OK**. The new Forcing Function/Measure Options window closes.
4. Save the change by clicking **Save** in the File menu, by clicking the single floppy-disk button in the toolbar, or by pressing [`Ctrl + S`] at the keyboard.

Figure 6-121 shows the result of the reassignment that was made per Figure 6-120 to the **charg_char** ITM.

Figure 6-121
New ITM after forcing function reassignment



Note that a newly assigned forcing function is configured with defaults that may not meet your needs. You may need to reconfigure the newly assigned forcing function, as discussed generally under Configuring SMU Forcing Functions/Measure Options window later in this section.

Reassigning forcing functions for an existing library ITM

The principles for reassigning forcing functions for an existing library ITM are identical to the principles previously discussed in Assigning forcing functions for a completely new ITM, except that the existing/default forcing function may not be the **Common** forcing function.

CAUTION	**Before reassigning a forcing function to an ITM, make sure that you want the change, which effectively creates a different ITM.**

Configuring SMU Forcing Functions/Measure Options window

A Forcing Functions/Measure Options window is associated with an instrument object that is assigned to a device terminal. The Forcing Functions/Measure Options window is used to configure the selected forcing function and measurements implemented by the instrument. This subsection helps you to do the following:

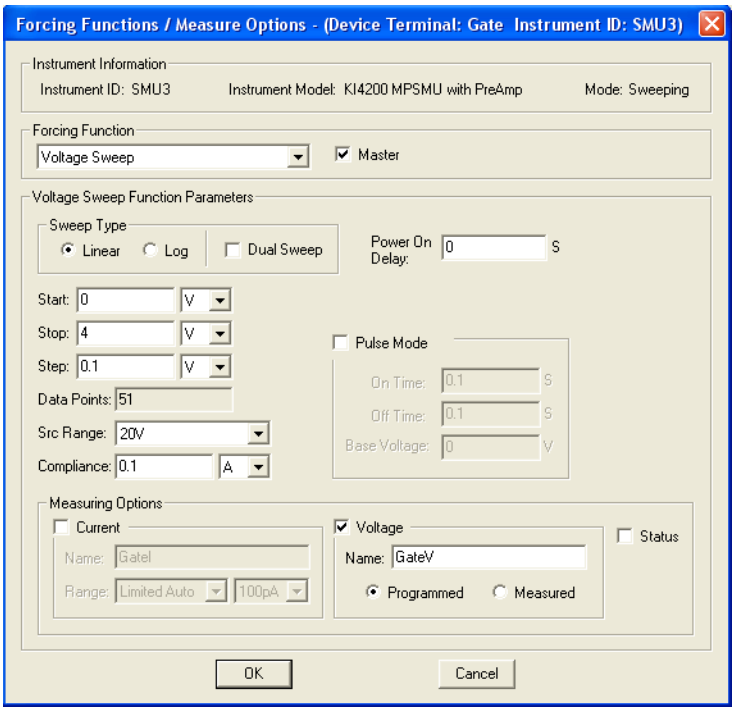
- Open a Forcing Functions/Measure Options window.
- Understand and configure each of the ten available forcing functions.
- Understand and configure the measurement options that are associated with a forcing function (Discussed generically, for all forcing functions, at the end of this subsection. Refer to Understanding and configuring the Measuring Options area later in this section).
- Understand and configure the other controls on the Forcing Functions/Measure Options window.

Opening a Forcing Functions/Measure Options window

To open a Forcing Function/Measure Options window for a terminal, first open the **Definition** tab for the ITM (refer to Opening an ITM window earlier in this section). Then, at the displayed terminal, click the **FORCE MEASURE** button of the associated instrument object. The Forcing Function/Measure Options window for that instrument object appears.

Figure 6-122 shows an existing Forcing Functions/Measure Options window that illustrates typical window features.

Figure 6-122
Forcing Functions/Measure Options window for an existing library ITM



Reviewing a typical Forcing Functions/Measure Options window

A typical Forcing Functions/Measure Options window, as shown in Figure 6-122, is divided as follows:

- The **Instrument Information** area.
- The **Forcing Function** area.
- The **<ForcingFunctionName> Function Parameters** area.
- The **Measuring Options** area (absent in the **Open** and **Common** windows).

Each of these areas is discussed below under a separate heading. Configuration parameters are explained in detail for the **Function Parameters** and **Measuring Options** areas.

Understanding the Instrument Information area

The display-only **Instrument Information** area of a Forcing Functions/Measure Options window summarizes the following selections in the ITM **Definition** tab:

- Information about the SMU selected for the corresponding device terminal.
- The test mode of the test being performed by the ITM (**Sweeping** mode or **Sampling** mode).

Understanding the Forcing Function area

Master checkbox

The **Master** checkbox is used to specify whether a current/voltage sweep, list sweep, or step forcing function acts as a master or slave.

For the significance of master and slave forcing functions, refer to one of the following subsections under The ForcingFunctionName function parameters area later in this section:

- Understanding Master Sweeps vs. Slave Sweeps
- Understanding Master List Sweeps vs. Slave List Sweeps
- Understanding Master Steps vs. Slave Steps

For clarifications of sweep, list sweep, or step forcing functions, refer to one of the following subsections under The ForcingFunctionName function parameters area:

- Understanding Linear Current and Voltage Sweeps
- Understanding log Current and Voltage Sweeps
- Understanding List Sweeps in general
- Understanding Step forcing functions vs. Sweep forcing functions

Forcing Function combo box

When you open a default Forcing Functions/Measure Options window for a device terminal, the default forcing function for that terminal appears in the **Forcing Function** combo box. One of the following ten available forcing functions, listed below by category, is displayed:

- Static (unswept/unstepped) forcing-functions
  - The **Open** forcing function
  - The **Common** forcing function
  - The **Current Bias** forcing function
  - The **Voltage Bias** forcing function

- Sweep forcing-functions (displayed only in the **Sweeping** test mode)
  - The **Current Sweep** forcing function
  - The **Voltage Sweep** forcing function
- List-sweep forcing functions (displayed only in the **Sweeping** test mode)
  - The **Current List Sweep** forcing function
  - The **Voltage List Sweep** forcing function
- Step forcing functions (displayed only in the **Sweeping** test mode)
  - The **Current Step** forcing function
  - The **Voltage Step** forcing function

NOTE *The Forcing Function combo box is used to assign, as well as display, forcing functions. Refer to Assigning/reassigning forcing functions to the device terminals earlier in this section.*

Power On Delay

When an ITM test is run, the SMUs for the given test power-on in a specific sequence. The first SMU in the sequence powers-on immediately.

Power-on delays can be set between all the SMUs in the test. The set delay for an SMU occurs after it powers-on, assuming there is another SMU in the power-on sequence.

For example, assume there are three SMUs in a test and the power-on sequence is SMU1, SMU2 and SMU3. Also assume that the power-on delay for SMU1 is set to 50 ms and the delay for SMU2 is set for 100 ms. When the test is started, the power-on sequence for the SMUs will be as follows:

SMU1 powers-on > 50 ms delay > SMU2 powers-on > 100 ms delay > SMU3 powers on

For Each SMU in the test, the **Power On Delay** field in the **Forcing Functions / Measure Options** window is used to set this delay, which can be set from 0 to 0.100 s.

The power-on sequence for the SMUs is set from the **ITM Timing** window that is launched by clicking the **Timing** button at the top of the ITM **Definition** tab. See Timing window later in this section for details.

Pulse Mode

To avoid device overheating in some tests, voltages or currents can be applied to a device only for brief periods at widely spaced intervals. For sweep (linear, log and list) and bias forcing functions, an SMU can be set to provide pulse output. With **Pulse Mode** enabled, the following pulse parameters can be set: **On Time**, **Off Time** and **Base Voltage** (or **Base Current**). The base is the level the SMU goes to between sweep points. Pulse on and off times determine pulse period and pulse width as follows:

Pulse period **= On Time** + **Off Time** + cumulative measure time (if set to measure)

Pulse width = **On Time**

Pulse Mode can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to **AUTO**.

Pulse on and off times can be set from 5 ms to 10 s. The base voltage (or current) that can be set is dependent the present source range (**Src Range**).

An example pulse output for the **Voltage Bias** forcing function is shown in Figure 6-123. Pulse output goes to the specified pulse level during the pulse on time. If the SMU is set to measure, the measurement will occur after the on time expires and before the transition to the off time level. This effectively increases the on time by the amount of time required to make the measurement.

Minimize this extra time by choosing 'custom' in the timing tab and setting delay and filter factor to 0, and A/D Integration factor to 0.01. This is the fastest (but least accurate) measurement timing scheme. If not set to measure, the output will simply transition from on to off.

During pulse off time, the pulse output returns to the specified **Base Voltage** level. After the off time expires, the output returns to 0 V.

For a sweep forcing function, pulse output steps to the sweep step levels during the pulse on times. During the off times, pulse output returns to the specified **Base Voltage** (or **Base Current**) level. Figure 6-124 shows an example of a single sweep and a dual sweep with the SMU set to measure. If not set to measure, the output will simply transition from the on levels to the off levels. The single sweep halts after reaching the **Stop** step. With **Dual Sweep** enabled, the test continues by sweeping from the **Stop** level back to the **Start** level. For details, see Understanding a Dual Sweep later in this section.

NOTE *More than one SMU in the test can be pulsing. If the pulse "on" or "off" times for the SMUs are different, the longer "on" and "off" times will take precedence in order for the SMUs to operate in a synchronized manner and run at the same speed.*

Figure 6-123
Pulse Mode example: Voltage bias; 2V level, 1V base

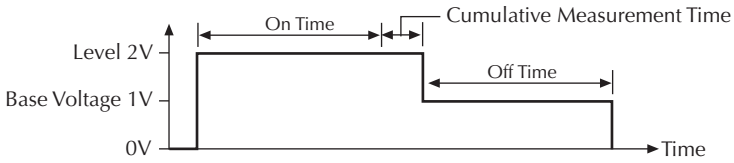
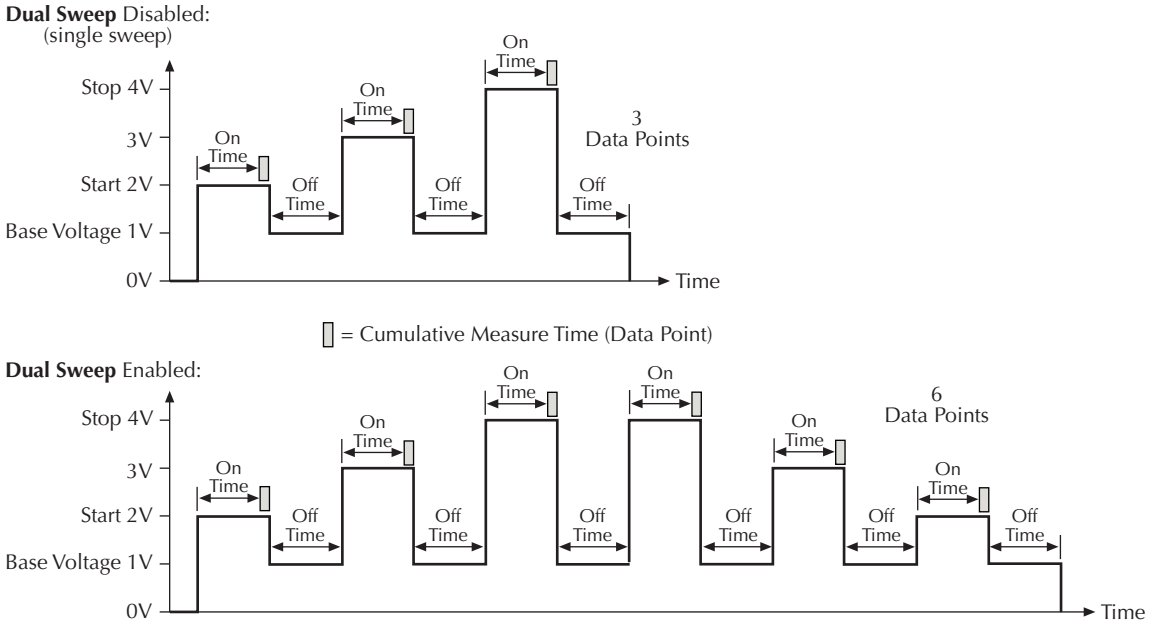


Figure 6-124
Pulse Mode examples: Single and dual sweep



The ForcingFunctionName function parameters area

This subsection provides the following:

- Explains each of the ten forcing functions, with amplification and examples as needed.
- Shows an example of each of the ten Forcing Functions/Measure Options windows.
- Discusses the parameters for each forcing function.

If you wish to make no changes in the **Forcing Function** combo box and retain a default Forcing Functions/Measure Options window, this subsection helps you to understand and configure the default function.

If you wish to create a custom ITM, the **Forcing Function** combo box allows you to select one of the alternative forcing functions (and an alternative Forcing Functions/Measure Options window) for the device terminal (refer to Assigning/reassigning forcing functions to the device terminals earlier in this section). This subsection helps you to understand each function before making a selection and then to configure the function that you select.

Static forcing functions

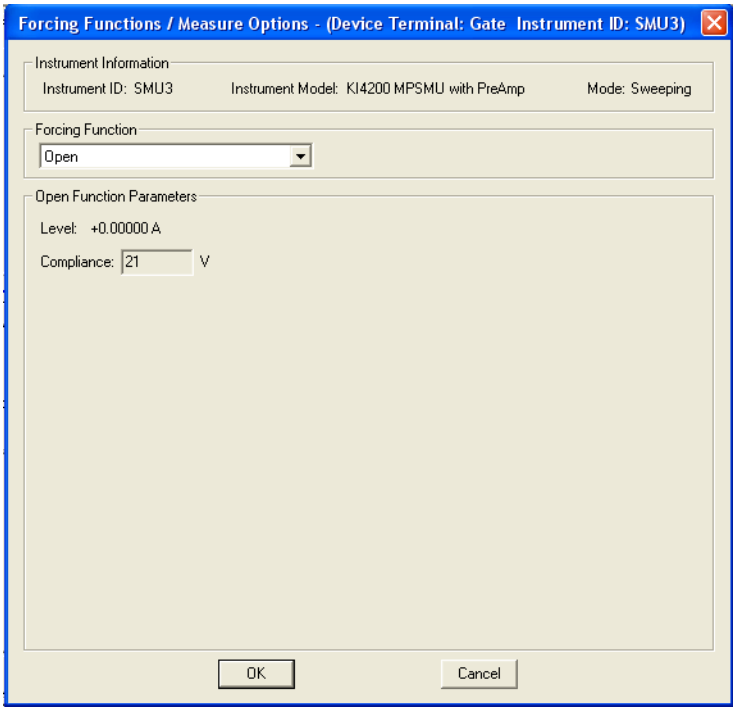
Each of the four static forcing functions applies a nondynamic (unswept/unstepped) condition at a device terminal. The static condition is typically applied for one of the following reasons:

- To maintain a fixed condition at one terminal of a device while sweeping or stepping another terminal(s) of the device (refer also to Sweep forcing functions later in this section).
- To maintain fixed conditions at device terminals while measuring a parameter change vs. time in the ITM **Sampling** mode (refer also to Selecting the ITM test mode earlier in this section).

Understanding and configuring the Open forcing function

The **Open** forcing function maintains a zero-current state at the terminal, subject to the maximum voltage compliance of the connected SMU. A typical **Open** Forcing Functions/Measure Options window is shown in Figure 6-125.

Figure 6-125
Open Forcing Functions/Measure Options window

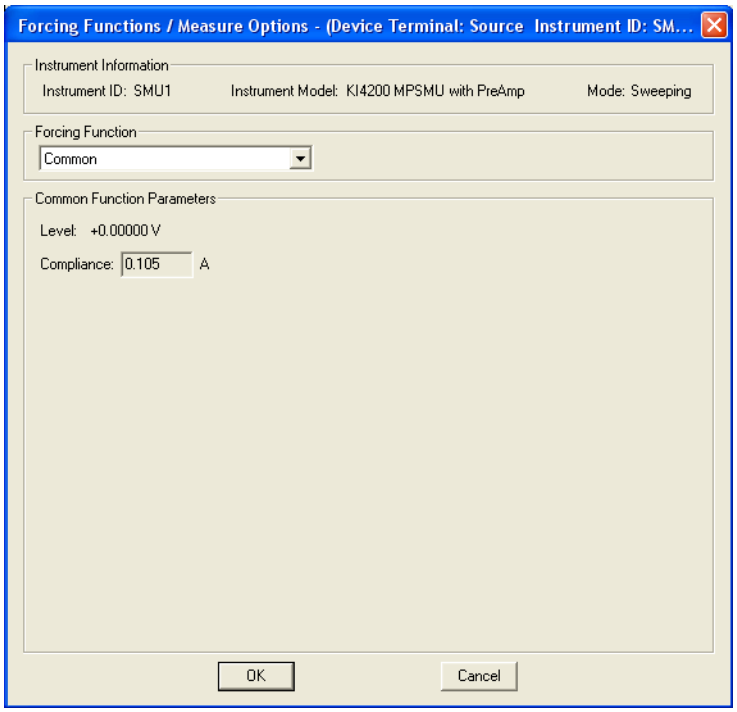


No function parameters are user configurable for the **Open** forcing function.

Understanding and configuring the Common forcing function

The **Common** forcing function maintains a zero-voltage state at the terminal, subject to the maximum current compliance of the connected SMU. A typical **Common** Forcing Functions/ Measure Options window is shown in Figure 6-126.

Figure 6-126
Common Forcing Functions/Measure Options window



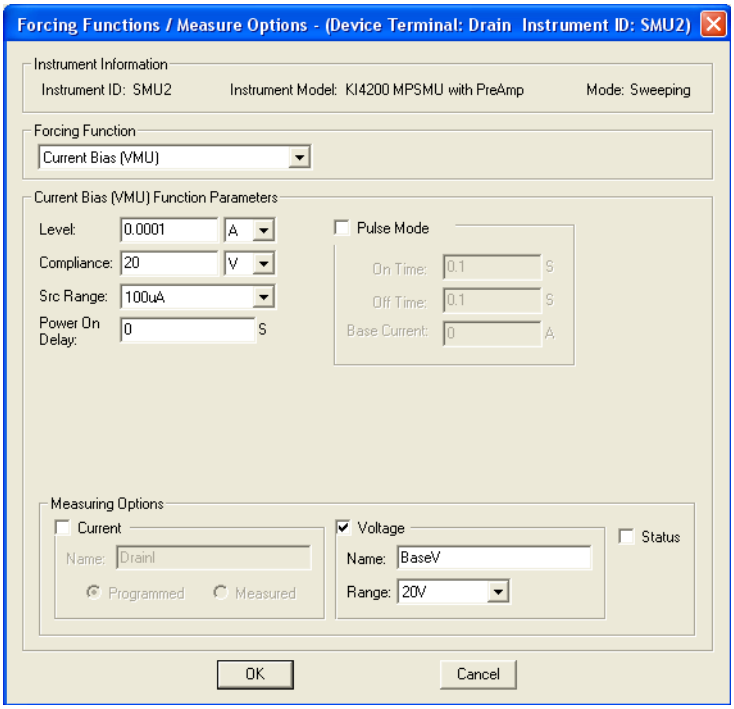
NOTE *Compliance can be 105 mA or 1.05 A, depending on whether the SMU is a Model 4200-SMU or 4210-SMU, respectively.*

No function parameters are user configurable for the **Common** forcing function.

Understanding and configuring the Current Bias forcing function

The **Current Bias** forcing function maintains a constant current state at the terminal, subject to the maximum voltage compliance of the connected SMU. A typical **Current Bias** Forcing Functions/ Measure Options window is shown in Figure 6-127.

Figure 6-127
Current Bias Forcing Functions/Measure Options window



The **Current Bias (VMU) Function Parameters** are configurable as follows:

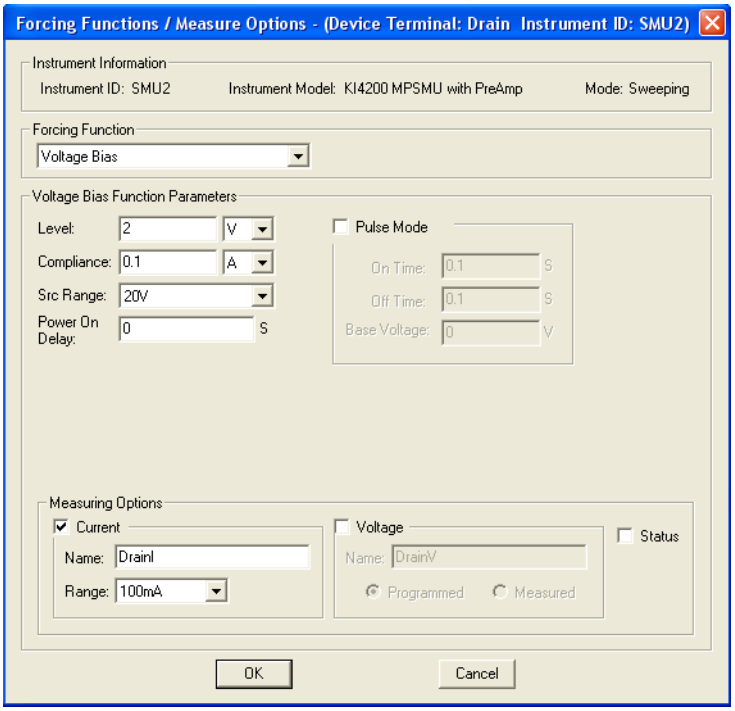
- The **Level** parameter edit box specifies the SMU current to be forced, the units for that are selected through the combo box at the right of the edit box.
- The **Compliance** parameter edit box specifies any valid SMU voltage compliance (or limit), the units for which are specified through the combo box at the right of the edit box.
- The **Src Range** (source range) combo box specifies the SMU current range used to force the bias current. For example, when forcing 30nA you would typically choose the 100nan SMU range or the **Best Fixed** SMU range, which allows the SMU to select the most appropriate range, based on the forcing value.
- **Power On Delay**: When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay earlier in this section.
- When selected, the **Pulse Mode** combo box allows definition of pulse on and off times, and pulse base current. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the specified current level during the pulse on time. During the off time, pulse output returns to the specified **Base Current** level. For details, see Pulse Mode earlier in this section.

NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

Understanding and configuring the Voltage Bias forcing function

The **Voltage Bias** forcing function maintains a selected constant-voltage state at the terminal, subject to a user-specified current compliance of the connected SMU. A typical **Voltage Bias** Forcing Functions/Measure Options window is shown in Figure 6-128.

Figure 6-128
Voltage Bias Forcing Functions/Measure Options window



The **Voltage Bias Function Parameters** are configurable as follows:

- The **Level** parameter edit box specifies any valid SMU voltage, the units for which are selected through the combo box at the right of the edit box.
- The **Compliance** parameter edit box specifies any valid SMU current compliance, the units for which are selected through the combo box at the right of the edit box.
- The **Src Range** (source range) combo box specifies the SMU voltage range used to force the bias voltage. For example, when forcing 5 V you would typically choose the 20 V SMU range or the **Best Fixed** SMU range, which allows the SMU to select the most appropriate range, based on the voltage level.
- **Power On Delay:** When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay earlier in this section.
- When selected, the **Pulse Mode** combo box allows definition of pulse on and off times, and pulse base voltage. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the specified voltage level during the pulse on time. During the off time, pulse output returns to the specified **Base Voltage** level. For details, see Pulse Mode earlier in this section.

NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

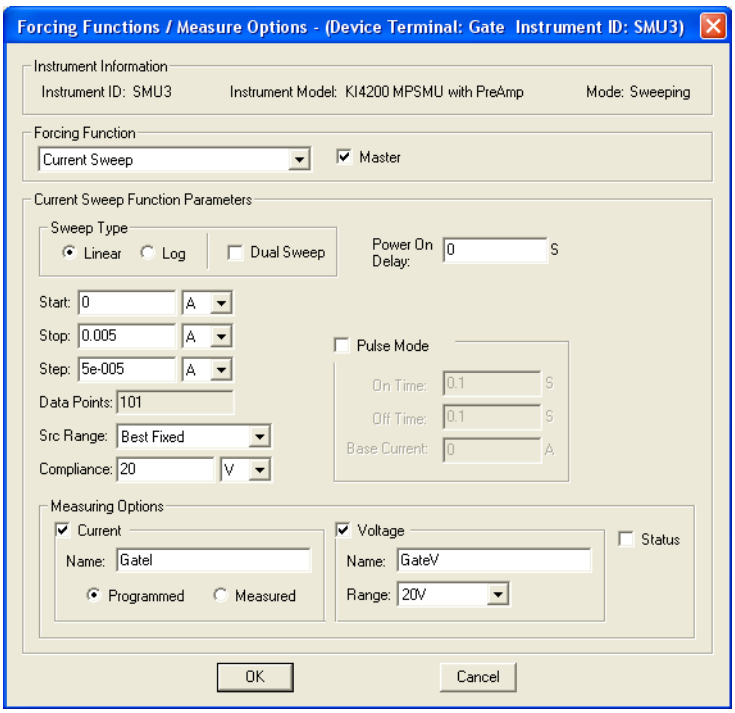
Sweep forcing functions

A sweep forcing function increments current or voltage at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. This sweep generates parametric curve data that is recorded in the ITM **Sheet** tab **Data** worksheet and can be plotted in the ITM **Graph** tab.

Understanding and configuring the Current Sweep forcing function

More specifically, the **Current Sweep** forcing function increments through a series of current steps over a user-specified range, subject to a user-specified voltage compliance. In a **Linear** sweep, the user-specified step size is uniform. In a **Log** sweep the KITE calculated step size is determined logarithmically. A typical **Current Sweep** Forcing Functions/Measure Options window is shown in Figure 6-129.

Figure 6-129
Current Sweep Forcing Functions/Measure Options window



The **Current Sweep Function Parameters** are configurable as listed below.

- **Sweep Type:** The **Sweep Type** radio button options, **Linear** and **Log**, specify whether the sweep is to be incremented linearly or in logarithmically calculated steps. These sweep options are explained subsequently under Understanding Linear Current and Voltage Sweeps and Understanding log Current and Voltage Sweeps later in this section.
- **Start:** The **Start** edit box specifies the current forced for the first data point of the sweep. The **Start** parameter may be any valid SMU current, the units for which are selected through the combo box at the right of the edit box.
- **Stop**: The **Stop** edit box determines the current forced for the last data point of the sweep. The **Stop** parameter may be any valid SMU current, the units for which are selected through the combo box at the right of the edit box.
- **Step (Linear** sweep case**)**: If **Sweep Type** is set to **Linear,** the **Step** edit box specifies the size of the current increments and determines the KITE calculated **Data Points** value {**Data points** value = integer value of [1 + (**Stop** value - **Start** value)/(**Step** value)]}.
- For linear sweeps, the **Step** parameter may be any valid SMU current value, the units for which are selected through the combo box at the right of the edit box. However, note the following:
  - It is best to specify a **Step** value that divides evenly into (**Stop** value - **Start** value). If the ratio [(**Stop** value - **Start** value)/(**Step** value)] is not an integer, the last, fractional step increment of the sweep is rejected and the last current value is smaller than the **Stop** value.

  - For example: If **Start** = 0A, **Stop** = 0.005 A, and **Step** = 0.0006 A:
    - **Data Points** value = integer of [1 + (0.005 - 0)/(0.0006)] = integer of [9.333] = 9
    - Nine values are forced: 0, 0.0006 A, 0.0012A, 0. 0018A,... 0.0042A, 0.0048A.
  - KITE never steps the force current beyond the value specified by the **Stop** parameter, even if you specify a **Step** value that is larger than the **Stop** value.
- **Step (Log** sweep case**)**: If **Sweep Type** is set to **Log,** the **Step** edit box is grayed and read-only. The value in the **Step** box (zero) has no significance, because the logarithmically calculated step sizes implemented by KITE are nonuniform (refer to Understanding log Current and Voltage Sweeps).
- **Data Points (Linear** sweep case**):** The **Data Points** value specifies the number of data points that are generated when a sweep is executed. If **Sweep Type** is set to **Linear,** the **Data Points** value is calculated by KITE and is read-only. Refer to the above description of the **Step** parameter.
  - **Data Points** value = integer of [1 + (**Stop** value - **Start** value)/(**Step** value)]
- **Data Points (Log** sweep case**):** The **Data Points** value specifies the number of data points that are generated when a sweep is executed. If **Sweep Type** is set to **Log,** a **Data Points** value between 1 and 4095 may be entered. For information about how KITE uses the **Data Points** value to calculate step size, refer to Understanding log Current and Voltage Sweeps.
- **Src Range**: The **Src Range** (source range) combo box specifies the SMU current range used to force the sweep current, per the following options:
  - The **Auto** option commands the SMU to automatically optimize the current range as the sweep progresses. This option provides the best current resolution and control when sweeping several decades. However, the range change time delays limit the sweep speed.
  - The **Best Fixed** option commands the SMU to automatically select the single current range that best fits the entire sweep.
  - The numerical current range options allow you to manually select an SMU range to suit your needs. This range must accommodate the **Stop** or **Start** value, whichever is greater.
- **Compliance**: The **Compliance** parameter edit box specifies a valid SMU voltage compliance for the sweep, the units for which are selected through the combo box at the right of the edit box.
- **Dual Sweep**: The SMUs in a test can be set to perform a dual sweep. When enabled, the SMU the will sweep from **Start** to **Stop**, and then continue to sweep from **Stop** back to **Start**. When disabled, the SMU will only sweep from **Start** to **Stop.** For details, see Understanding a Dual Sweep.
- **Power On Delay**: When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay.
- **Pulse Mode:** When selected, the **Pulse Mode** combo box allows definition of a current pulse sweep. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the sweep step levels during the pulse on times. During the off times, pulse output returns to the specified **Base Current** level. For details, see Pulse Mode.

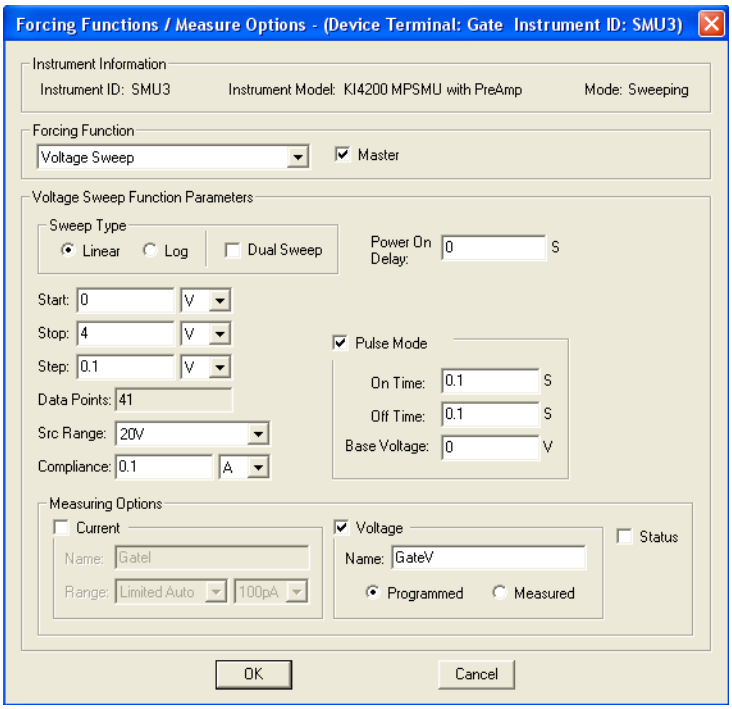
NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

Understanding and configuring the Voltage Sweep forcing function

The **Voltage Sweep** forcing function increments through a series constant voltage steps over a user-specified range, subject to a user-specified current compliance. In a **Linear** sweep, the user-specified step size is uniform. In a **Log** sweep the KITE calculated step size is determined

logarithmically. A typical **Voltage Bias** Forcing Functions/Measure Options window is shown in Figure 6-130.

Figure 6-130
Voltage Sweep Forcing Functions/Measure Options window



The **Voltage Sweep Function Parameters** are configurable in essentially the same way as the **Current Sweep Function Parameters**, as follows:

- **Sweep Type:** The **Sweep Type** radio button options, **Linear** and **Log**, specify whether the sweep is to be incremented linearly or in logarithmically calculated steps. These sweep options are explained subsequently under Understanding Linear Current and Voltage Sweeps and Understanding log Current and Voltage Sweeps later in this section.
- **Start:** The **Start** edit box specifies the voltage forced for the first data point of the sweep. The **Start** parameter may be any valid SMU voltage, the units for which are selected through the combo box at the right of the edit box.
- **Stop**: The **Stop** edit box determines the voltage forced for the last data point of the sweep. The **Stop** parameter may be any valid SMU voltage, the units for which are selected through the combo box at the right of the edit box.
- **Step (Linear** sweep case**)**: If **Sweep Type** is set to **Linear,** the **Step** edit box specifies the size of the voltage increments and determines the KITE calculated **Data Points** value {**Data points** value = integer value of [1 + (**Stop** value - **Start** value)/(**Step** value)]}.

For linear sweeps, the **Step** parameter may be any valid SMU voltage value, the units for which are selected through the combo box at the right of the edit box. However, note the following:

- It is best to specify a **Step** value that divides evenly into (**Stop** value - **Start** value). If the ratio [(**Stop** value - **Start** value)/(**Step** value)] is not an integer, the last, fractional step increment of the sweep is rejected and the last voltage value is smaller than the **Stop** value.
- For example: If **Start** = 0 V, **Stop** = 5 V, and **Step** = 0.6 V:
  - **Data Points** value = integer of [1+ (5 - 0)/(0.6)] = integer of [9.333] = 9
  - Nine values are forced: 0 V, 0.6 V, 1.2 V, 1.8 V,... 4.2 V, and 4.8 V.
- KITE never steps the force voltage beyond the value specified by the **Stop** parameter, even if you specify a **Step** value that is larger than the **Stop** value.

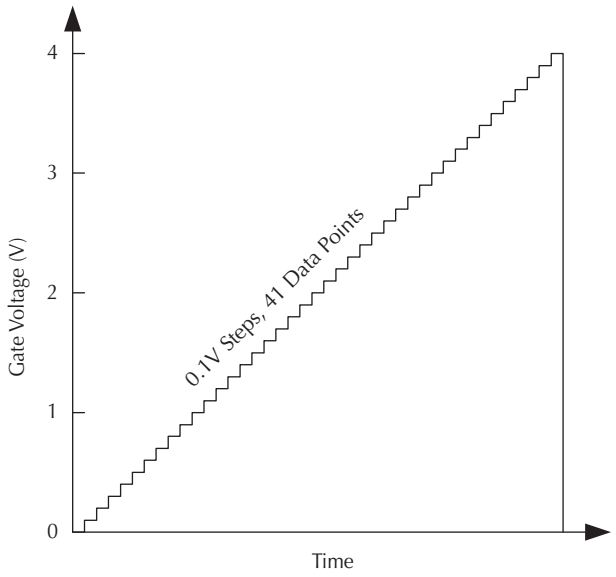
- **Step (Log** sweep case**)**: If **Sweep Type** is set to **Log,** the **Step** edit box is grayed and read-only. The value in the **Step** box (zero) has no significance, because the logarithmically calculated step sizes implemented by KITE are nonuniform (refer to Understanding log Current and Voltage Sweeps).
- **Data Points (Linear** sweep case**):** The **Data Points** value specifies the number of data points that are generated when a sweep is executed. If **Sweep Type** is set to **Linear,** the **Data Points** value is calculated by KITE and is read-only. Refer to the above description of the **Step** parameter.
  - **Data Points** value = integer of [1 + (**Stop** value - **Start** value)/(**Step** value)]
- **Data Points (Log** sweep case**):** The **Data Points** value specifies the number of data points that are generated when a sweep is executed. If **Sweep Type** is set to **Log,** a **Data Points** value between 1 and 4095 may be entered. For information about how KITE uses the **Data Points** value to calculate step size, refer to Understanding log Current and Voltage Sweeps.
- **Src Range**: The **Src Range** (source range) combo box specifies the SMU voltage range used to force the sweep voltage, per the following options:
  - The **Auto** option commands the SMU to automatically optimize the voltage range as the sweep progresses. This option provides the best voltage resolution and control when sweeping several decades. However, the range-change time delays limit the sweep speed.
  - The **Best Fixed** option commands the SMU to automatically select the single voltage range that best fits the entire sweep.
  - The numerical voltage range options allow you to manually select an SMU range to suit your needs. This range must accommodate the **Stop** or **Start** value, whichever is greater.
- **Compliance**: The **Compliance** parameter edit box specifies a valid SMU current compliance limit for the sweep, the units for which are selected through the combo box at the right of the edit box.
- **Dual Sweep**: The SMUs in a test can be set to perform a dual sweep. When enabled, the SMU the will sweep from **Start** to **Stop**, and then continue to sweep from **Stop** back to **Start**. When disabled, the SMU will only sweep from **Start** to **Stop.** For details, see Understanding a Dual Sweep.
- **Power On Delay**: When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay.
- **Pulse Mode:** When selected, the **Pulse Mode** combo box allows definition of a voltage pulse sweep. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the sweep step levels during the pulse on times. During the off times, pulse output returns to the specified **Base Voltage** level. For details, see Pulse Mode.

NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

Understanding Linear Current and Voltage Sweeps

The Forcing Function/Measure Options window in Figure 6-130 defines a **Linear** sweep. A linear sweep increments current or voltage in a series of uniform steps, at a speed that is determined by the **Speed** and **Timing** settings (discussed subsequently under Configuring the Speed and Timing settings in the ITM Definition tab). Figure 6-131 is a graph of the linear sweep forcing function that is defined in Figure 6-130.

Figure 6-131
Linear Sweep Forcing Function example



Understanding log Current and Voltage Sweeps

Some applications require the following:

- The forcing parameter must be swept over a large range. Current, especially, may be swept over several decades.
- The graph of the forcing function must be plotted on a logarithmic scale.

A linear sweep is typically unsatisfactory for such applications, because the first increment can miss several of the lower decades. For example, the first ~0.1V **Step** of a 101-point linear sweep from 0.001V to 10 V jumps over the two decades between 0.001 volt and 0.1 volt.

By contrast, a **Log** sweep varies the **Step** size logarithmically over the specified range, so that all decades are characterized uniformly. Each point in the sweep is forced according to the following equations:

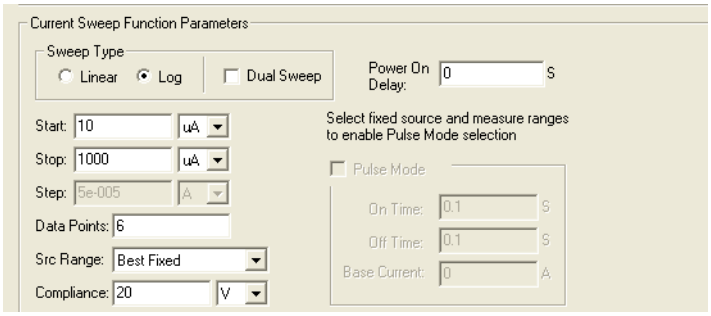
Step size = ($\log_{10}$ |**Stop** value| - $\log_{10}$ |**Start** value|) / (**Data Points** value - 1)

SMU forcing value for data point n = 10 [$\log_{10}$ |**Start** value| + (n-1)(**Step** size)]

The **Stop, Start,** and **Data Points** values in the above equations are as specified in the **Function Parameters** area of a **Current Sweep** or **Voltage Sweep** Forcing Functions/Measure Options window. For a log sweep, you specify the **Data Points** value and KITE calculates the **Step** size (the opposite is true for a linear sweep).

Specify a log sweep in the **Function Parameters** area of the window by clicking the **Log** button under **Sweep Type**. The **Function Parameters** area that then displays is slightly different than for linear sweeps. See Figure 6-132.

Figure 6-132
Log Sweep Function parameters

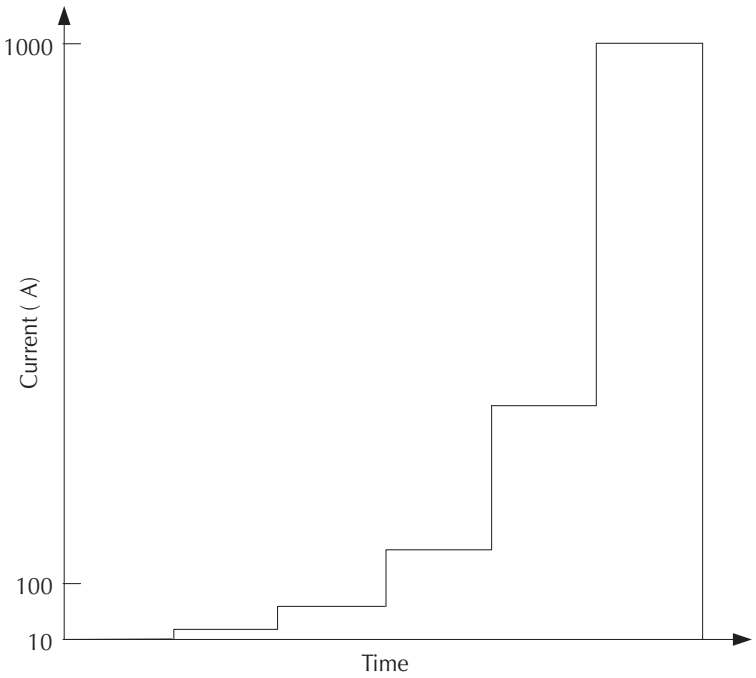


The log sweep function parameters are the same as the linear sweep function parameters. However, the parameters that are user-configurable differ.

NOTE *The **Start** value entered or a log sweep must not be zero (refer to the equations above).*

Figure 6-133 graphically illustrates the log sweep forcing function that is defined in Figure 6-132.

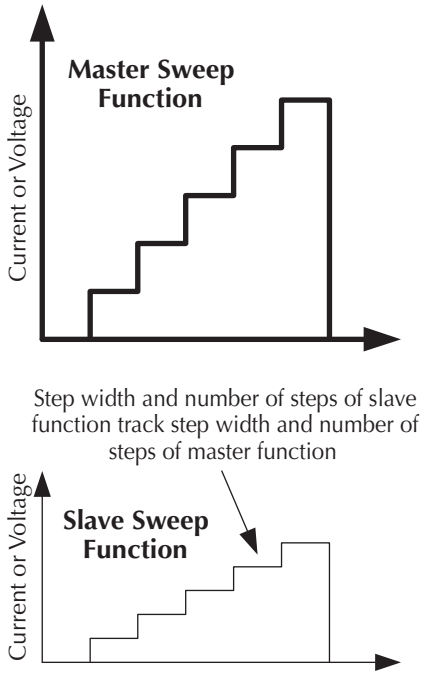
Figure 6-133
Log Sweep Forcing Function example



Understanding Master Sweeps vs. Slave Sweeps

It is possible to simultaneously apply multiple sweep forcing functions to multiple device terminals. However, all of the sweep forcing functions in the ITM must track with regard to increment number and duration. Therefore, one of the sweep functions must be user-designated as the **Master.** All other sweep functions, by default, are then automatically designated by KITE as slaves. Figure 6-134 illustrates this concept.

Figure 6-134
Master Sweep Function vs. Slave Sweep Function



To specify a particular sweep function as the master sweep function, click the **Master** checkbox in the **Forcing Function** area of its Forcing Functions/Measure Options window. If you do not specify a particular sweep function as the master step function, the first sweep function that you assign to an ITM is the master, by default.

In Forcing Functions/Measure Options windows, KITE enforces tracking relationships between master and slave sweep parameters as follows:

- **Linear sweep master/linear sweep slave(s)** or **Log sweep master/linear sweep slave(s):** KITE sets the slave **Data Points** value and **Step** size to be the same as the master **Data Points** value and **Step** size. User changes to the slave **Data Points** value are not allowed.
- **Log sweep master/log sweep slave(s)** or **Linear sweep master/log sweep slave(s)**: KITE sets the slave **Data Points** value and **Step** size to be the same as the master **Data Points** value and **Step** size. User changes to the slave **Data Points** value are ignored.

Understanding a Dual Sweep

A SMU that is configured to perform a linear or log sweep, can also be set to perform a dual sweep. With **Dual Sweep** enabled, the SMU will essentially perform two sweeps. The first sweep steps from the **Start** level to the **Stop** level. The SMU then continues with the second sweep, that steps from the **Stop** level back to the **Start** level. With **Dual Sweep** disabled, the SMU performs a single sweep stepping from **Start** to **Stop**.

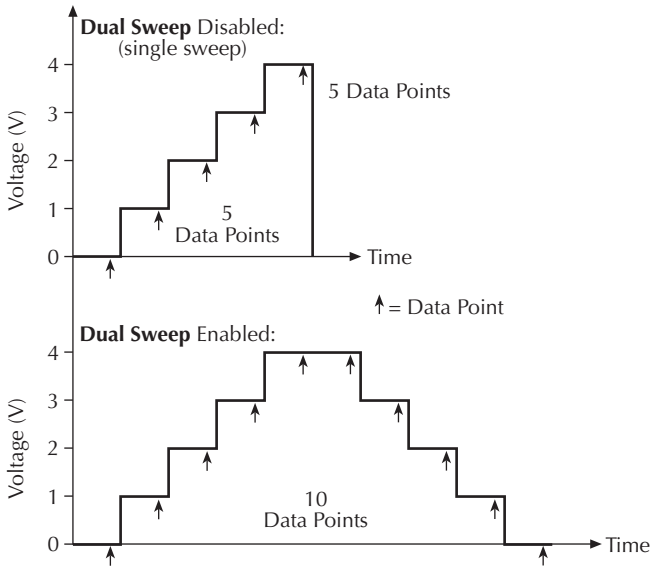
A dual sweep for a slave SMU is typically used in concert with a **Master** SMU that is also set to perform a dual sweep. The master SMU does not have to be set for dual sweep in order to use **Dual Sweep** for a slave SMU. In this case, setting the master SMU's sweep points to an even number will ensure that the slave's dual sweep is symmetric. Setting the master SMU count to an odd number, will cause the slave SMU to repeat the last sweep point.

The slaves SMUs WILL NOT automatically set for dual sweep when **Dual Sweep** is enabled for the master SMU. **Dual Sweep** must be individually enabled for each SMU.

Figure 6-135 compares a single sweep to a dual sweep.

NOTE *A dual sweep can also be performed with the **Pulse Mode** selected (see Pulse Mode).*

Figure 6-135
Single and dual sweep examples (linear voltage sweep; 0 to 4V in 1V steps)



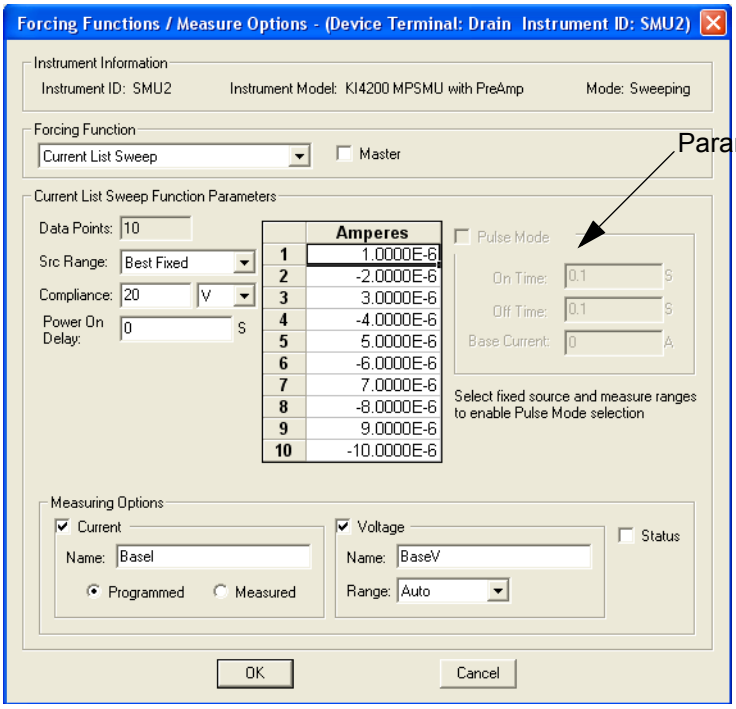
List sweep forcing functions

A list sweep forcing function steps a current or voltage through a list of user-specified values, at a rate that is determined by the **Timing** and **Speed** settings in the ITM **Definition** tab. This *sweep* generates parametric data that is recorded in the ITM **Sheet** tab **Data** worksheet and can be plotted in the ITM **Graph** tab, if desired.

Understanding and configuring the Current List Sweep forcing function

The **Current List Sweep** forcing function increments through a series current values, each of which is individually user-specified. Almost any pattern of steps may be configured, subject to the range limitations of the SMU and the specified voltage compliance. Refer to the illustrations in Understanding List Sweeps in general. A typical **Current List Sweep** Forcing Functions/Measure Options window is shown in Figure 6-136.

Figure 6-136
Current List Sweep Forcing Functions/Measure Options window



The **Current List Sweep Function Parameters** are configurable as listed below:

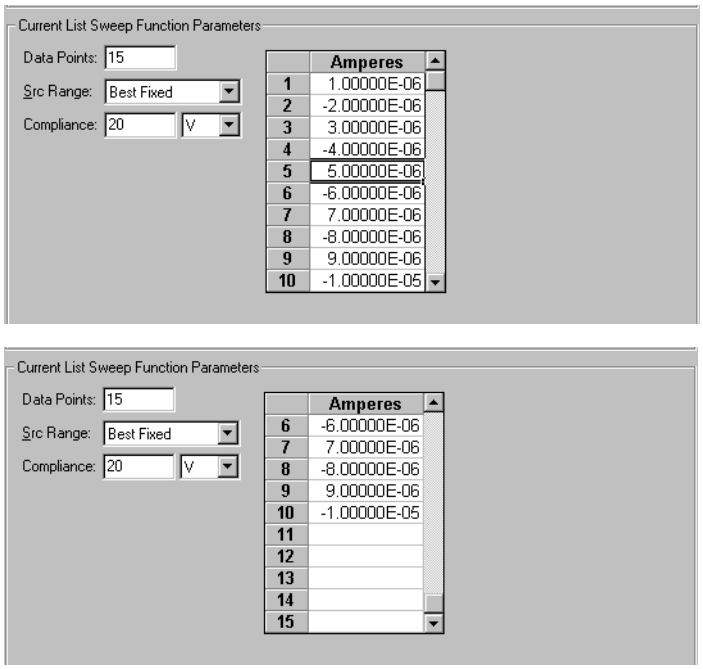
- **Parameter list:** The parameter list specifies the sequence and value of each current to be forced during the list sweep. Each parameter value may be any valid SMU current.

NOTE *KITE requires all parameter list cells to be filled before you leave the Forcing Functions/Measure Options window.*

- **Data Points:** The **Data Points** value, an integer between 0 and 4095, specifies the following:
  - The number of user-specified current values in the parameter list (refer to next bulleted item).
  - The number of data points that are generated when a sweep is executed.
- The default **Data Points** value is 10.

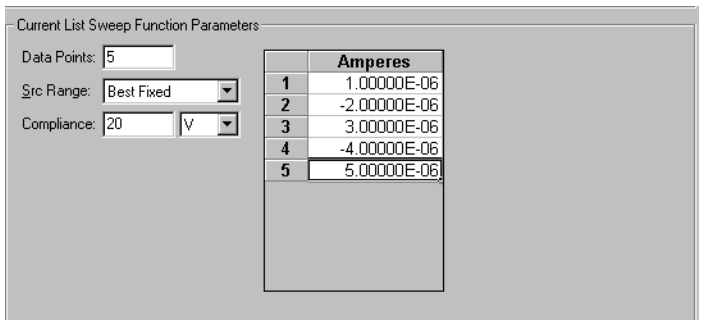
- If you increase the **Data Points** value to greater than 10, the parameter list changes to a scroll list when you select any cell. See Figure 6-137.

Figure 6-137
Results of increasing the Data Points value beyond the default of 10



- If you decrease the **Data Points** value from any prior size, say from size N to size M, the parameter list shrinks; this can be seen when you select any list cell. Any prior parameter values with indices higher than the **Data Points** value (that is, prior parameter values numbered M+1, M+2, ..., N-1, N-2) are *discarded*. Figure 6-138 illustrates the results of decreasing the **Data Points** value from 10 to 5 or from 15 to 5.

Figure 6-138
Results of decreasing the Data Points value



- **Src Range**: The **Src Range** (source range) combo box specifies the SMU current range that is used to force the sweep currents, per the following options:
  - The **Auto** option commands the SMU to automatically change the current range to the optimal range as the sweep progresses. This option provides the best current resolution and control when sweeping several decades. However, the range-change time delays limit the sweep speed.
  - The **Best Fixed** option commands the SMU to automatically select the single current range that best fits the entire sweep.

- The numerical current range options allow you to manually select a fixed SMU range to suit your needs. This range must be equal to or greater than the largest absolute value in the parameter list.

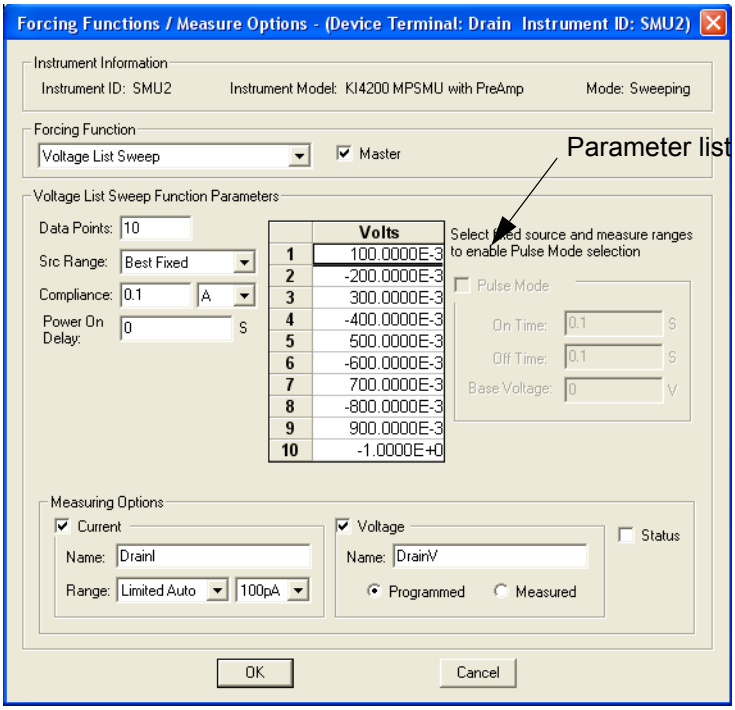
- **Compliance**: The **Compliance** parameter edit box specifies any valid SMU voltage compliance limit, the units for which are selected through the combo box at the right of the edit box.
- **Power On Delay**: When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay.
- **Pulse Mode:** When selected, the **Pulse Mode** combo box allows definition of a current pulse list sweep. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the sweep list levels during the pulse on times. During the off times, pulse output returns to the specified **Base Current** level. For details, see Pulse Mode.

NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

Understanding and configuring the Voltage List Sweep forcing function

The **Voltage List Sweep** forcing function increments through a series of voltage values, each of which is individually user-specified. Almost any pattern of steps may be configured, subject to the range limitations of the SMU and the specified current compliance. Refer to the illustrations in Understanding List Sweeps in general. A typical **Voltage Sweep** Forcing Functions/Measure Options window is shown in Figure 6-139.

Figure 6-139
Voltage List Sweep Forcing Functions/Measure Options window



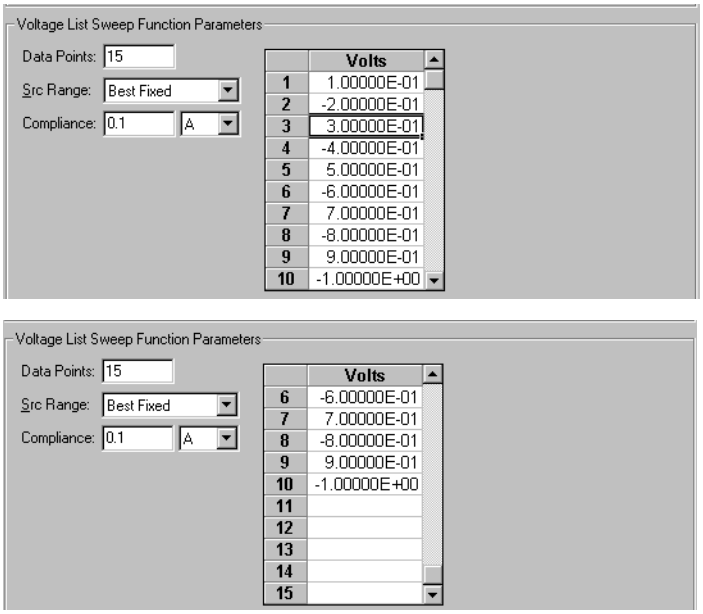
The **Voltage List Sweep Function Parameters** are configurable as listed below:

- **Parameter list:** The parameter list specifies the sequence and value of each voltage to be forced during the list sweep. Each parameter value may be any valid SMU voltage.

NOTE *KITE requires all parameter list cells to be filled before you leave the Forcing Functions/Measure Options window.*

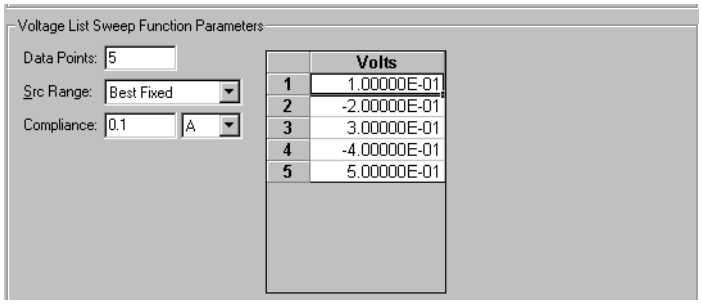
- **Data Points:** The **Data Points** value, an integer between 0 and 4095, specifies the following:
  - The number of user-specified voltage values in the parameter list (refer to next bulleted item).
  - The number of data points that are generated when a sweep is executed.
- The default **Data Points** value is 10.
  - If you increase the **Data Points** value to greater than 10, the parameter list adds a scroll bar, which can be seen when you select any cell. See Figure 6-140.

Figure 6-140
Results of increasing the Data Points value beyond the default of 10



  - If you decrease the **Data Points** value from any prior size, say from size N to size M, the parameter list shrinks; this can be seen when you select any list cell. Any prior parameter values with indices higher than the **Data Points** value (that is, prior parameter values numbered M+1, M+2, ..., N-1, N-2) are *discarded*. Figure 6-141 illustrates the results of decreasing the **Data Points** value from 10 to 5 or from 15 to 5.

Figure 6-141
Results of decreasing the Data Points value



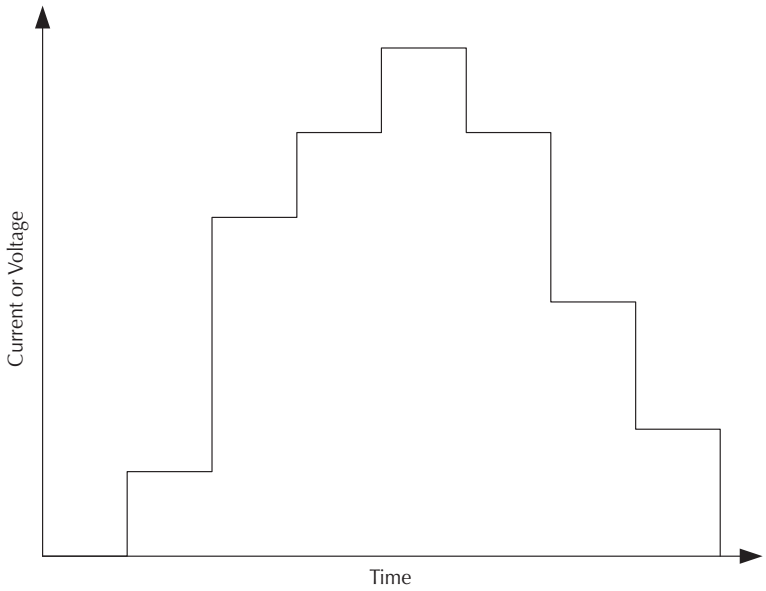
- **Src Range**: The **Src Range** (source range) combo box specifies the SMU voltage range that is used to force the sweep voltages, per the following options:
  - The **Auto** option commands the SMU to automatically change to the optimal range as the sweep progresses. This option provides the best voltage resolution and control when sweeping several decades. However, the range change time delays limit the sweep speed.
  - The **Best Fixed** option commands the SMU to automatically select the single voltage range that best fits the entire sweep. This range is based on the absolute value of the maximum voltage.
  - The numerical voltage range options allow you to manually select a fixed SMU range to suit your needs. This range must be equal to or greater than the largest value in the parameter list.
- **Compliance**: The **Compliance** parameter edit box specifies any valid SMU current compliance limit, the units for which are selected through the combo box at the right of the edit box.
- **Power On Delay**: When an ITM test is run, the SMUs for the given test power-on in a specific sequence. A power-on delay can be set between the SMUs in the test. The **Power On Delay** field is used to set this delay, which can be set from 0 to 0.100 s. For details, see Power On Delay.
- **Pulse Mode:** When selected, the **Pulse Mode** combo box allows definition of a voltage pulse list sweep. The on and off times can be set from 5 ms to 10 s. Pulse output goes to the sweep list levels during the pulse on times. During the off times, pulse output returns to the specified **Base Voltage** level. For details, see Pulse Mode.

NOTE ***Pulse Mode*** *can be selected ONLY when source and measure ranges are fixed. In other words, Pulse Mode is disabled if the source or measure range is set to* ***AUTO****.*

Understanding List Sweeps in general

List Sweeps allow you to make measurements only at selected forced voltages and currents. For example, they allow you to skip unimportant measurement points or to synthesize a custom sweep that is based on a special mathematical equation (as described in the next subsection, list sweeps also allow you to make pulsed measurements to avoid overheating of sensitive devices). Figure 6-142 illustrates a possible list sweep.

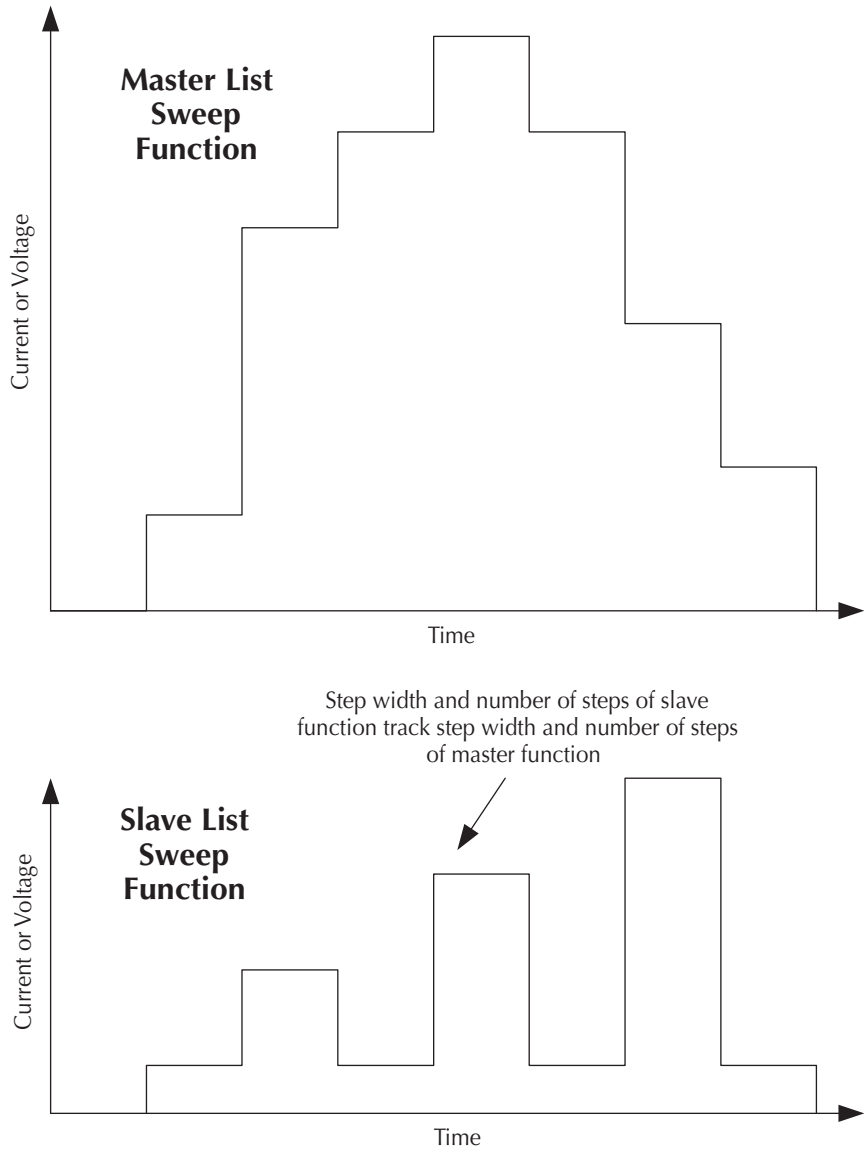
Figure 6-142
List Sweep Function general illustration



Understanding Master List Sweeps vs. Slave List Sweeps

It is possible to simultaneously apply multiple list sweep forcing functions to multiple device terminals. However, all of the list sweep forcing functions in the ITM must track with regard to increment number and duration. Therefore, one of the list sweep functions must be designated by the user as the **Master**. All other list sweep functions are automatically designated by KITE as slaves. Figure 6-143 illustrates this concept.

Figure 6-143
Master Lists Sweep Function vs. Slave List Sweep Function



To specify a particular list sweep function as the master list-sweep function, click the **Master** checkbox in the **Forcing Function** area of its Forcing Functions/Measure Options window. If you do not specify a particular list sweep function as the master list sweep function, the first list-sweep function that you assign to an ITM is the master, by default.

In Forcing Functions/Measure Options windows, KITE enforces tracking between master list-sweep and slave list sweep user entries is as follows:

- When a slave list sweep function is assigned to the ITM (by default, after the master function):
  - The number of cells in a slave function parameter list is fixed by KITE to correspond to the number of cells in the master function parameter list.
  - KITE requires all slave function parameter list cells to be filled in before leaving the Forcing Functions/Measure Options window.
- If the number of cells in the master function parameter list is decreased, the following occurs:
  - The number of cells in a slave function parameter list is reduced by KITE to correspond to the number of items in the cells in the master function parameter list.
  - The current/voltage values that were in the removed cells are discarded.
- If the number of cells in the master function parameter list is increased, the following occurs:
  - The number of cells in a slave function parameter list is increased by KITE to correspond to the number of items in the cells in the master function parameter list.
  - KITE requires all added slave function parameter list cells to be filled in before leaving the Forcing Functions/Measure Options window.

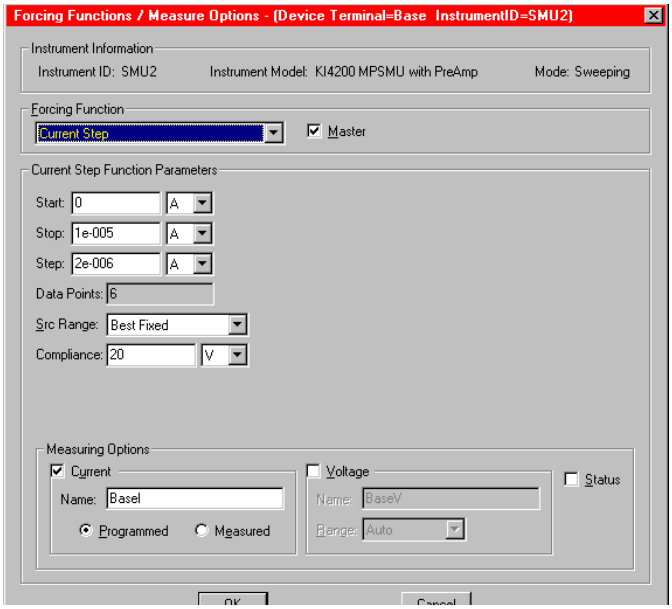
Step forcing functions

A step forcing function incrementally steps a current or voltage to two or more levels, each of which is held constant during the progress of a voltage or current sweep at another terminal (refer to Sweep forcing functions). For each step, parametric curve data is recorded in the ITM **Sheet** tab **Data** worksheet. The combined data can be plotted in the ITM **Graph** tab, resulting in a series (family) of curves.

Understanding and configuring the Current Step forcing function

More specifically, the **Current Step** forcing function increments through multiple evenly spaced, constant current steps over a user-specified range, subject to a user-specified voltage compliance. The time interval for each step is determined automatically by the time required to complete a sweep. A typical **Current Step** Forcing Functions/Measure Options window is shown in Figure 6-144.

Figure 6-144
Current Step Forcing Functions/Measure Options window



The **Current Step Function Parameters** are configurable as listed below.

- **Start:** The **Start** edit box specifies the current forced for the first step value. The **Start** parameter may be any valid SMU current, the units for which are selected through the combo box at the right of the edit box.
- **Stop**: The **Stop** edit box determines the current forced for the last step value. The **Stop** parameter may be any valid SMU current, the units for which are selected through the combo box at the right of the edit box.
- **Step**: The **Step** edit box specifies the current-step increments and determines the **Data Points** value {**Data points** value = integer value of [1 + (**Stop** value - **Start** value)/(**Step** value)].
  The **Step** parameter may be any valid SMU current value, the units for which are selected using the combo box at the right of the edit box. However, note the following:
  - It is best to specify a **Step** value that divides evenly into (**Stop** value - **Start** value). If the ratio [(**Stop** value - **Start** value)/(**Step** value)] is not an integer, the last, fractional step increment of the sweep is rejected and the last current value is smaller than the **Stop** value.
  - For example: If **Start** = 0A, **Stop** = 0.005 A, and **Step** = 0.0015 A:
    - **Data Points** value = integer of [1+ (0.005 - 0)/(0.0015)] = integer of [4.333] = 4
    - Four values are forced: 0A, 0.0015 A, 0.0030A, and 0. 0045 A.
  - KITE never steps the force current beyond the value specified by the **Stop** parameter, even if you specify a **Step** value that is larger than the **Stop** value.
- **Data Points:** The **Data Points** value is calculated by KITE and is read-only. Refer to the above description of the Step parameter.
  - **Data Points** value = integer of [1 + (**Stop** value - **Start** value)/(**Step** value)]
- **Src Range**: The **Src Range** (source range) combo box specifies the SMU current range used to force the sweep current, per the following options:
  - The **Auto** option commands the SMU to automatically optimize the current range as the stepping progresses. This option provides the best current resolution and control when stepping several decades. However, the range-change time delays limit the stepping speed.

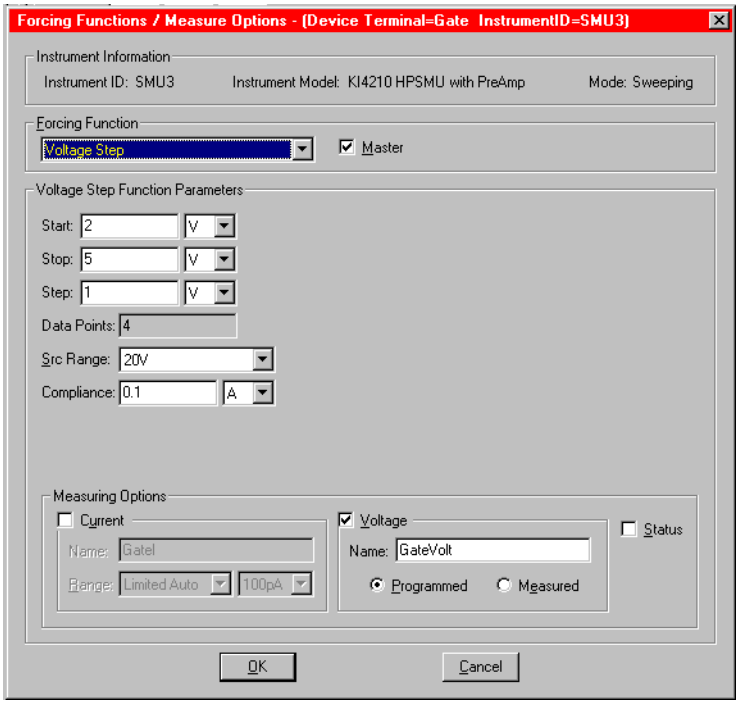
- The **Best Fixed** option commands the SMU to automatically select the single current range that best fits the entire step range.
- The numerical current range options allow you to manually select an SMU range to suit your needs.

- **Compliance**: The **Compliance** parameter edit box specifies any valid SMU voltage compliance limit, the units for which are selected through the combo box at the right of the edit box.

Understanding and configuring the Voltage Step forcing function

The **Voltage Step** forcing function increments through a multiple evenly spaced, constant voltage steps over a user-specified range, subject to a user-specified current compliance. The time interval for each step is determined automatically by the time required to complete a sweep. A typical **Voltage Step** Forcing Functions/Measure Options window is shown in Figure 6-145.

Figure 6-145
Voltage Step Forcing Functions/Measure Options window



The **Voltage Step Function Parameters** are configurable in essentially the same way as the **Current Sweep Step Parameters**, as follows:

- **Start:** The **Start** edit box specifies the voltage forced for the first step value. The **Start** parameter may be any valid SMU voltage, the units for which are selected through the combo box at the right of the edit box.
- **Stop**: The **Stop** edit box determines the voltage forced for the last step value. The **Stop** parameter may be any valid SMU voltage, the units for which are selected through the combo box at the right of the edit box.
- **Step**: The **Step** edit box specifies the voltage step increments and determines the **Data Points** value {**Data points** value = integer value of [1 + (**Stop** value - **Start** value)/(**Step** value)].
  The **Step** parameter may be any valid SMU voltage value, the units for which are selected through the combo box at the right of the edit box. However, note the following:

- It is best to specify a **Step** value that divides evenly into (**Stop** value - **Start** value). If the ratio [(**Stop** value - **Start** value)/(**Step** value)] is not an integer, the last, fractional step increment of the sweep is rejected and the last voltage value is smaller than the **Stop** value.
- For example: If **Start** = 0 V, **Stop** = 5 V, and **Step** = 0.6V:
  - **Data Points** value = integer of [1+ (5 - 0)/(0.6)] = integer of [9.333] = 9
  - Nine values are forced: 0 V, 0.6V, 1.2V, 1.8V,... 4.2V, 4.8V
- KITE never steps the force voltage beyond the value specified by the **Stop** parameter, even if you specify a **Step** value that is larger than the **Stop** value.

- **Data Points:** The **Data Points** value is calculated by KITE and is not directly user-configurable. Refer to the above description of the Step parameter.
  - **Data Points** value = integer of [1 + (**Stop** value - **Start** value)/(**Step** value)]
- **Src Range**: The **Src Range** (source range) combo box specifies the SMU voltage range used to force the sweep voltage, per the following options:
  - The **Auto** option commands the SMU to automatically optimize the voltage range as the stepping progresses. This option provides the best voltage resolution and control when stepping several decades. However, the range-change time delays limit the step speed.
  - The **Best Fixed** option commands the SMU to automatically select the single voltage range that best fits the entire step range.
  - The numerical voltage range options allow you to manually select an SMU range to suit your needs.
- **Compliance**: The **Compliance** parameter edit box specifies any valid SMU current compliance, the units for which are selected through the combo box at the right of the edit box.

Understanding Step forcing functions vs. Sweep forcing functions

Figure 6-146 illustrates how one **Definition** tab, for the **"vds-id"** ITM, specifies both Step and Sweep forcing functions.

Figure 6-146
Example Definition tab that includes both stepping and sweeping

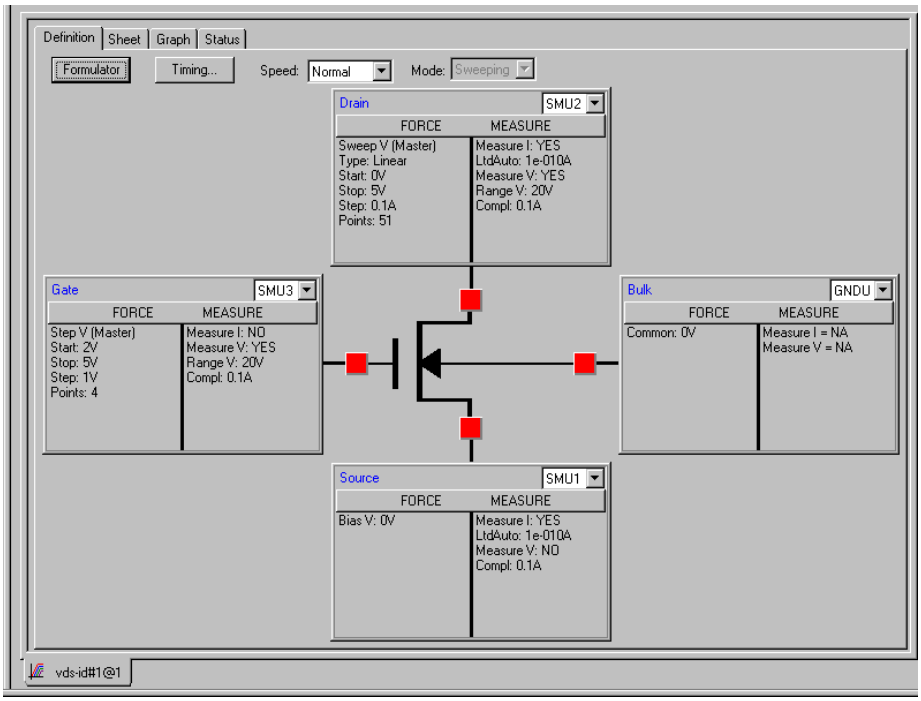
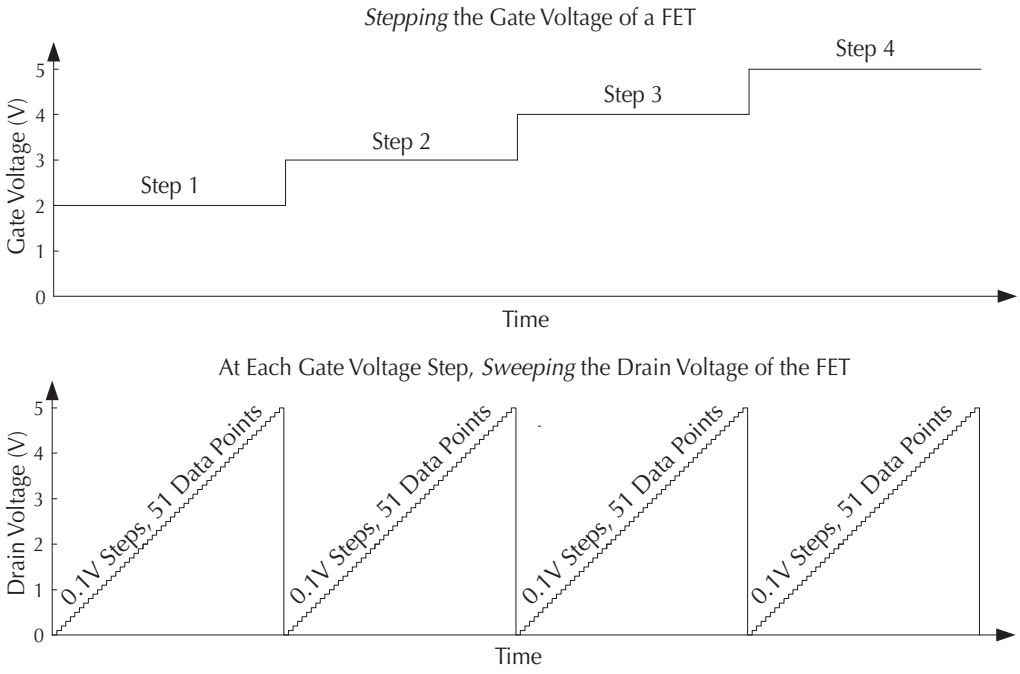


Figure 6-147 graphically illustrates the combined Step and Sweep forcing functions specified in Figure 6-146.

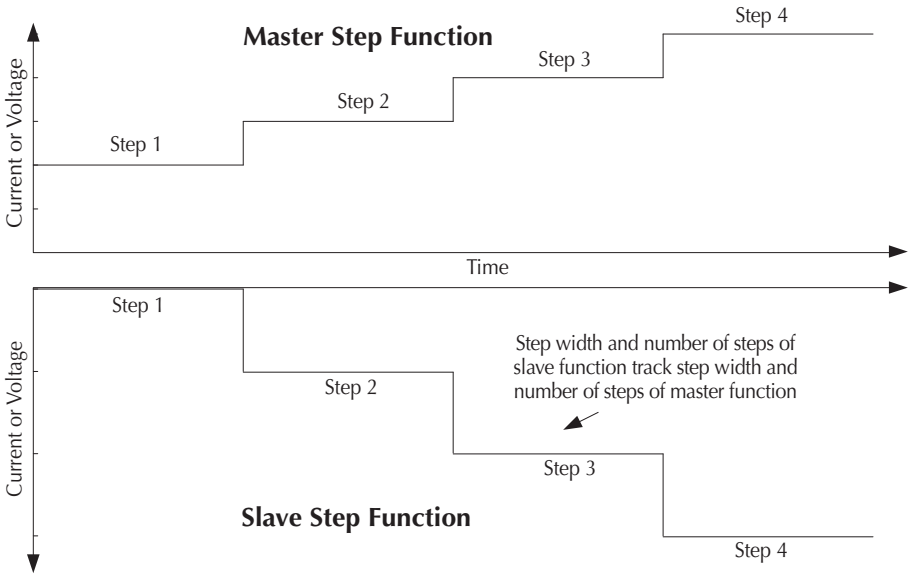
Figure 6-147
Stepping and sweeping example



Understanding Master Steps vs. Slave Steps

It is possible to simultaneously apply multiple step forcing functions to multiple device terminals (for example, stepping the biases on two transistor terminals and sweeping voltage or current on the third terminal). However, all of the sweep forcing functions in the ITM must track with regard to step number and duration. Therefore, one of the step functions must be user-designated as the **Master.** All other step functions are automatically designated as slaves. Figure 6-148 illustrates this concept.

Figure 6-148
Master Step function vs. Slave Step function



To specify a particular step function as the master step function, click the **Master** checkbox in the **Forcing Function** area of its Forcing Functions/Measure Options window. If you do not specify a particular step function as the master step function, the first step function that you assign to an ITM is the master function, by default.

To enforce tracking, KITE sets the slave **Data Points** value and **Step** size to be the same as the master **Data Points** value and **Step** size. User changes to the slave **Data Points** value are not allowed in the Forcing Functions/Measure Options window.

Understanding and configuring the Measuring Options area

The **Measuring Options** area shown in Figure 6-149 and Figure 6-150 together display all of the options that you may encounter when configuring an ITM (forcing Function/Measure Options windows for voltage and current forcing functions display some different options, and windows for the **Common** and **Open** forcing functions display no options).

Figure 6-149
Typical Measuring Options area for a voltage forcing function

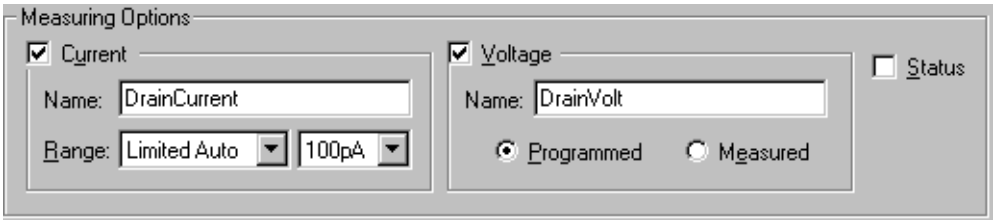
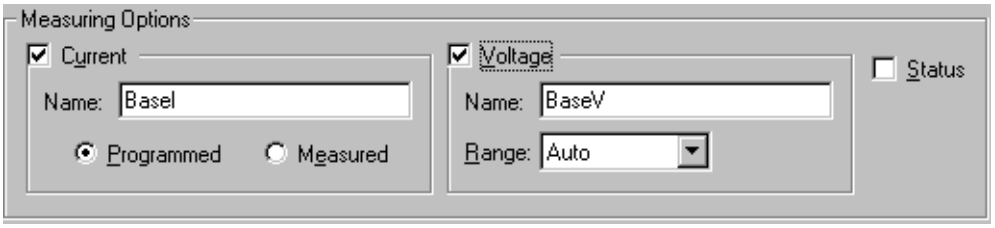


Figure 6-150
Typical Measuring Options area for a current forcing function



Current measuring options

Current checkbox

Check the **Current** checkbox if you desire the current to be recorded in the **Sheet** tab **Data** worksheet and be available for plotting in the **Graph** tab. If you do not check the **Current** checkbox, KITE disables current measurements for the corresponding device terminal and grays the current measuring options.

Current Name edit box

In the current **Name** edit box, you can specify a name to be given to the current measurement in a **Sheet** tab column and, if you plot it, on a **Graph** tab axis (you can later rename a Graph axis as desired).

Otherwise, KITE assigns a default name, which is a combination of the terminal label (for example, Base, Anode, or for a capacitor or resistor, A or B) and I for current. The following are examples of default current names: BaseI, AnodeI, or AI.

Current Range combo box

The current **Range** (measure range) combo box is present for voltage forcing functions only. Here you specify the measurement range to be used by the SMU for current, from the following options:

- The **Auto** option commands the SMU to automatically optimize the current measurement range as stepping/sweeping progresses. This option provides the best resolution when the measurements span several decades. However, range-change time delays limit the measurement speed.
- The **Limited Auto** option is a compromise between the full **Auto** option and a fixed range option. Here you can specify the minimum range that the SMU uses when automatically optimizing the current measurements. This option saves time when you do not need maximum resolution at minimum currents.
- The **Best Fixed** option commands the SMU to automatically select a single measurement range, based on the current compliance.
- The numerical range options allow you to manually select a fixed current measurement range to suit your needs.

Programmed and Measured radio buttons

The *current* **Programmed** and **Measured** radio buttons are present for current forcing functions only. They specify the current values that are recorded in the **Sheet** tab **Data** worksheet, as follows.

- Clicking the **Programmed** radio button specifies that the as-requested forced current values are to be recorded in the **Sheet** tab **Data** worksheet. For example, under the **Programmed** option, if you specify in a **Current Bias** window that 10 mA is to be forced at

a transistor collector terminal, the reported value is exactly 10 mA, even if the measured value would be 9.9982mA.

- Clicking the **Measured** radio button specifies that the current values to be recorded in the **Sheet** tab **Data** worksheet are actual measured values (by way of the SMU A/D converter). For example, under the **Measured** option, if you specify in a **Current Bias** window that 10 mA is to be forced at a transistor collector terminal, the reported value is 9.9982mA if the measured value is 9.9982mA.

NOTE *The **Measured** mode essentially doubles the measurement time, because of the need for an extra analog-to-digital (A/D) conversion.*

Voltage measuring options

Voltage checkbox

Check the **Voltage** checkbox if you desire the voltage to be recorded in the **Sheet** tab **Data** worksheet and be available for plotting in the **Graph** tab. If you do not check the **Voltage** checkbox, KITE disables voltage measurements for the corresponding device terminal and grays the voltage measuring options.

Name edit box

In the voltage **Name** edit box, you can specify a name to be given to the voltage measurement in a **Sheet** tab column and, if you plot it, on a **Graph** tab axis (You can later rename a Graph axis as desired).

Otherwise, KITE assigns a default name, which is a combination of the terminal label (for example, Drain, Anode, or (for a capacitor or resistor) A or B) and V for voltage. The following are examples of default voltage names: DrainV, AnodeV, BV.

Voltage Range combo box

The voltage **Range** (measure range) combo box is present for current forcing functions only. Here you specify the measurement range to be used by the SMU for voltage, from the following options:

- The **Auto** option commands the SMU to automatically optimize the voltage measurement range as stepping/sweeping progresses. This option provides the best resolution when the measurements span multiple decades. However, range-change time delays limit the measurement speed.
- The **Best Fixed** option commands the SMU to automatically select a single measurement range, based on voltage compliance.
- The numerical range options allow you to manually select a fixed voltage measurement range to suit your needs.

Voltage Programmed and Measured radio buttons

The voltage **Programmed** and **Measured** radio buttons are present for voltage forcing functions only. They specify the voltage values that are recorded in the **Sheet** tab **Data** worksheet, as follows.

- Clicking the **Programmed** radio button specifies that as-requested forced voltage values are to be recorded in the **Sheet** tab **Data** worksheet. For example, under the **Programmed** option, if you specify in a **Voltage Bias** window that 2.5 V is to be forced at a transistor drain terminal, the reported value is exactly 2.5 V, even if the measured value would be 2.4997V.
- Clicking the **Measured** radio button specifies that the voltage values to be recorded in the **Sheet** tab **Data** worksheet are actual measured values (by way of the SMU A/D converter).

For example, under the **Measured** option, if you specify in a **Voltage Bias** window that 2.5 V is to be forced at a transistor drain terminal, the reported value is 2.4997V if the measured value is 2.4997V.

NOTE *The **Measured** mode essentially doubles the measurement time, because of the need for an extra analog-to-digital (A/D) conversion.*

Status checkbox

If you check the **Status** checkbox on a Forcing Functions/Measure Options window, KITE records measurement status information when the ITM executes. KITE records a four hexadecimal byte (8 nibble, 32-bit binary) SMU status code in the **Sheet** tab **Data** worksheet for each set of SMU measurements. This code, variable between 0000 and FF8F, reports the forcing mode (voltage or current), voltage and current ranges, and overflow and compliance status.

Table 6-7 explains the groups of code bits.

Table 6-7
ITM status-code bit map

Bit	Summary Description	Details
31	Reserved	Reserved bits are reserved for future use. The state of reserved status bits is undefined and is not guaranteed under any conditions, unless specifically indicated.
30		
29		
28		
27		
26		
25		
24		
23		
22		
21		
20		
19		
18		
17		
16		
15	Reserved; always 0	
14		
13		
12		
11	Reserved	
10		
9		
8		
7		
6		
5		
4		

Table 6-7 (continued)
ITM status-code bit map

Bit	Summary Description	Details
3	Overflow	1 if the measurement circuit was overloaded when the measurement was made, 0 otherwise. For example, measuring 5 Vwhile on the 2V range causes this bit to be 1.
2	Range compliance	1 if the SMU was in range compliance when the measurement was made or if the programmed compliance limit was reached during the measurement, 0 otherwise. For example, assume the SMU is sourcing voltage and measuring current on the 1 mA range, and the programmed current compliance limit is 60 mA. Because the SMU is fixed on the 1 mA range, it will enter range compliance when the measured current attempts to exceed 1 mA; the range-compliance bit will be set accordingly.
1	Compliance	1 only if the SMU reached the programmed compliance limit during the measurement, 0 otherwise.
0	Source mode	1 if the SMU was programmed to force current during the measurement, 0 if the SMU was programmed to force voltage.

Configuring the Speed and Timing settings in the ITM Definition tab

Two key issues in making good measurements with a low-current semiconductor analyzer are addressed by way of controls on the ITM **Definition** tab (settling time and noise).

- **Settling time**: The settling time is the time that a measurement takes to stabilize after the voltage or current is changed, such as during execution of a Sweep forcing function. Settling time includes the following τ (tau) time constants:
  - $^{\tau}$Instrument: Varies mainly with current range
  - $^{\tau}$System: Due to cables/switches/probers
  - $^{\tau}$DUT (Device Under Test): Due to the implicit characteristics of the DUT
  - $^{\tau}$Dielectric Absorption: An issue only on the low current ranges
- **Noise**: The noise on a measurement is affected by many factors, but the basic relationship is this: the larger the A/D integration time (the longer the A/D converter looks at the signal), the lower the noise. A/D integration times are usually configured in Number of Power Line Cycles (N:PLCs), defined as follows:
  - For a 60Hz power line, one PLC time = 16.6ms.
  - For a 50Hz power line, one PLC time = 20 ms.

  Power lines are principal sources of noise. Integration of power line noise over precisely one or more full cycles cancels this noise.
- At the lower current ranges, you generally need relatively long A/D integration times to reduce the noise and provide acceptable signal-to-noise ratios. Conversely, at the higher ranges, you can generally achieve acceptable signal-to-noise ratios using shorter A/D integration times.

Settling time and noise are addressed by the following controls on an ITM **Definition** tab:

- **Speed** combo box.
- **Timing** button, which opens the Timing Panel.

These controls are discussed below, under Speed combo box and Timing window.

Speed combo box

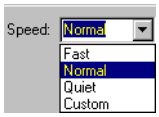
The Model 4200-SCS is highly tuned to automatically take into account both settling time and noise issues. It provides four measurement-speed modes that allow the user to select trade-offs between speed and noise. These modes, **Fast**, **Normal**, **Quiet**, and **Custom**, are selectable directly in the **Speed** combo box, as well as in the ITM Timing window (discussed in the next subsection):

- **Fast**: Optimizes the Model 4200-SCS for speed at the expense of noise performance. It is a good choice for fast and dirty measurements where noise and settling time are not concerns.
- **Normal**: The default and most commonly used setting. It provides a good combination of speed and low noise and is the best setting for most cases.
- **Quiet**: Optimizes the Model 4200-SCS for low-noise measurements at the expense of speed. If speed is not a critical consideration, it is a good choice when you need the lowest noise and most accurate measurements.
- **Custom**: Allows you to fine tune the timing parameters to meet a particular need, using the Timing window (discussed in the next subsection). With **Custom**, you can configure the A/D integration time and individual delay and filtering factors to produce a composite setting that is faster than the **Fast** setting, quieter than the **Quiet** setting, or anything in between.

Select **Fast**, **Normal**, **Quiet**, or **Custom** directly from the **Speed** combo box as follows:

1. In the ITM **Definition** tab, click the arrow button on the **Speed** combo box. The scroll list shown in Figure 6-151 appears.

Figure 6-151
Speed scroll list



2. Click the desired setting.

After selecting the measurement **Speed** mode, do one of the following:

- If you selected the **Fast**, **Normal**, or **Quiet** measurement **Speed** mode, you *may* not need to further configure timing settings. However, some settings in the Timing window apply to the **Fast**, **Normal**, and **Quiet** measurement **Speed** modes, especially if you are making measurements in the **Sampling** test mode. To understand those settings and which may be important for your application, review the next subsection, Timing window later in this section.
- If you selected the **Custom** measurement **Speed** mode, continue with the next subsection, Timing window.

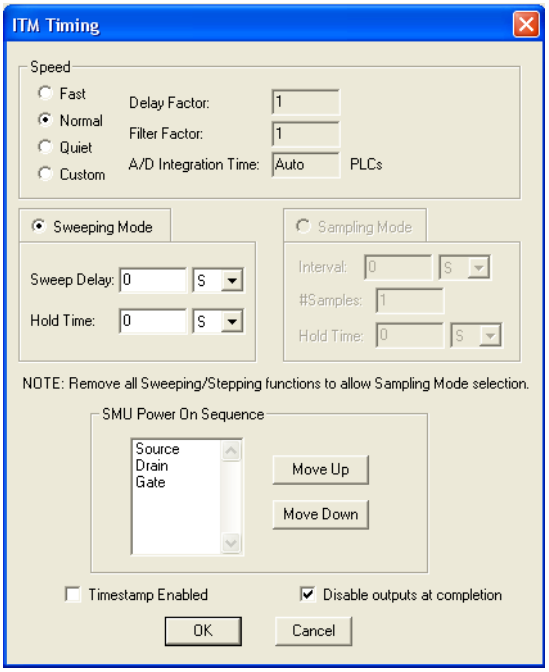
Timing window

The Timing window is used to configure ITM timing settings for a Model 4200-SCS SMU. Three of the settings can be configured only in the **Custom** measurement **Speed** mode. The others can be configured in all measurement **Speed** modes. In summary, you can do the following in the Timing window:

- Locally select the **Fast**, **Normal**, **Quiet**, or **Custom** measurement **Speed** mode (also selectable directly as described under Speed combo box).
- Configure custom **Delay Factor**, **Filter Factor**, and **A/D Integration Time** settings (only in the **Custom** measurement **Speed** mode).
- Add special delays for **Sweeping** test-mode tests or configure the timing for **Sampling** test-mode tests.
- Set the SMU power-on sequence when a test is started.
- Enable a timestamp to be recorded for each measurement.

To access the ITM timing parameters, click the **Timing** button at the top of the ITM **Definition** tab. The ITM Timing window appears, as shown in Figure 6-152.

Figure 6-152
ITM Timing window



NOTE *Multiple measurement delays are configurable in the Timing window. However, if you are using autoranging, note that these delays do not include autoranging delays, which can be substantial.*

Understanding and configuring the Speed area of the Timing window

The **Speed** area of the Timing window allows you 1) to locally select one of the measurement-Speed modes and, 2) if you select the **Custom** measurement **Speed** mode, to also configure additional **Delay Factor**, **Filter Factor**, and **A/D Integration Time** settings.

Fast, Normal, Quiet, and Custom selections

You can locally select the measurement **Speed** mode in the ITM Timing window by clicking the **Fast**, **Normal**, **Quiet**, or **Custom** radio button. These selections are identical to the selections in the **Speed** combo box (refer to Speed combo box later in this section).

Delay Factor setting

After applying a forced voltage or current, an SMU waits for a delay time before making a measurement. The delay time allows for source settling. The default delay time is pre-programmed and range-dependent, to allow for the very long settling times needed at very low current ranges. The *applied* delay time is a multiple of the default delay time, and the value in the **Delay Factor** edit box specifies this multiple. That is:

Applied delay time = (Default delay time) × (**Delay Factor**)

For example, if the default delay time is 1ms and the **Delay Factor** is 0.7, the applied delay time is 0.7ms (1ms × 0.7).

If you select the **Custom** measurement **Speed** mode, you can enter a custom **Delay Factor** of 0 to 100. If you select the **Fast**, **Normal,** or **Quiet** measurement **Speed** mode, the SMU chooses an appropriate, fixed **Delay Factor**. Table 6-8 summarizes allowed **Delay Factor** settings for various measurement **Speed** modes.

Table 6-8
Summary of allowed Delay Factor settings

Speed Mode	Delay Factor Settings
Fast	0.7
Normal	1.0
Quiet	1.3
Custom	0 to 100

When entering a custom **Delay Factor** setting, consider the following:

- A **Delay Factor** of 1 allows for a normal amount of settling before the A/D converter is triggered to make a measurement.
- Each doubling of the **Delay Factor** doubles the time allowed for settling.
- A delay factor of 0 multiplies the default delay by zero, resulting in no delay.

Filter Factor setting

To reduce measurement noise, each Model 4200-SCS SMU applies filtering, which may include averaging of multiple readings to make one measurement. The SMU automatically adjusts the filtering to fit the selected measurement range (this system works particularly well, because measurements at lower current ranges require much more filtering [and much more time] than at higher ranges). The value entered in the **Filter Factor** edit box specifies a multiple of this preprogrammed filtering.

If you select the **Custom** measurement **Speed** mode, you can enter a **Filter Factor** value of 0 to 100. If you select the **Fast**, **Normal**, or **Quiet** measurement **Speed** mode, the SMU sets an appropriate fixed **Filter Factor**. Table 6-9 summarizes allowed **Filter Factor** settings for various measurement **Speed** modes.

Table 6-9
Summary of allowed Filter Factor settings

Speed Mode	Filter Factor Settings
Fast	0.2
Normal	1
Quiet	3
Custom	0 to 100

When entering a custom **Filter Factor**, consider the following:

- A **Filter Factor** of 1 specifies a normal level of filtering.
- As a rule of thumb, doubling the **Filter Factor** halves the measurement noise.
- A **Filter Factor** of 0 nullifies the SMU internal filtering.

***A/D Integration Time** setting*

The entry in the **A/D Integration Time** edit box controls the A/D converter integration time used to measure a signal. Each measured reading by a Model 4200-SCS SMU is the result of one or more A/D (analog-to-digital) conversions. A short integration time for each A/D conversion results in a relatively fast measurement-speed at the expense of noise. A long integration time results in a relatively low noise reading at the expense of speed. The integration time setting is based on the number of power line cycles (NPLCs). For 60Hz line power, 1.0 PLC = 16.67msec (1/60). For 50Hz line power, 1.0 PLC = 20 msec (1/50).

A custom **A/D Integration Time** setting controls the A/D conversion time as follows:

- The word **Auto** in the **A/D Integration Time** edit box specifies that the Model 4200-SCS SMU unit picks the optimum A/D conversion time for most normal measurements.
- A numerical value in the **A/D Integration Time** edit box specifies the minimum A/D conversion time in units of Number of Power Line Cycles (NPLC). The applied A/D conversion time also depends on the Filter Factor setting, as described below:
  - If the **Filter Factor** setting is nonzero, the SMU chooses and applies an optimum A/D converter time that is based on the **Filter Factor** setting. The applied A/D converter time value is *never less* than the specified **A/D Integration Time.**
  - If the **Filter Factor** setting is zero, the SMU applies a fixed A/D converter time that equals the specified **A/D Integration Time**.

If you select the **Custom** measurement **Speed** mode, you can enter **Auto** or any value between 0.01 and 10 NPLC. If you select the **Fast**, **Normal** or **Quiet** measurement **Speed** mode, KITE sets the **A/D Integration Time** to **Auto**. Table 6-10 summarizes allowed **A/D Integration Time** settings for various measurement **Speed** modes.

Table 6-10
Summary of allowed A/D Integration Time settings

Speed Mode	A/D Integration Time Setting
Fast	Auto
Normal	Auto
Quiet	Auto
Custom	0.01 to 10 PLC

Understanding and configuring the Sweeping Mode area of the Timing window

If any terminal of the device under test is configured for a dynamic forcing function (a step or sweep forcing function) the **Sweeping Mode** is automatically enabled. Then, you can configure two **Sweeping Mode** settings in the Timing window: **Sweep Delay** and **Hold Time**. You can configure these settings for all measurement **Speed** modes, **Fast**, **Normal, Quiet,** and **Custom,** as discussed below.

Sweeping Mode settings

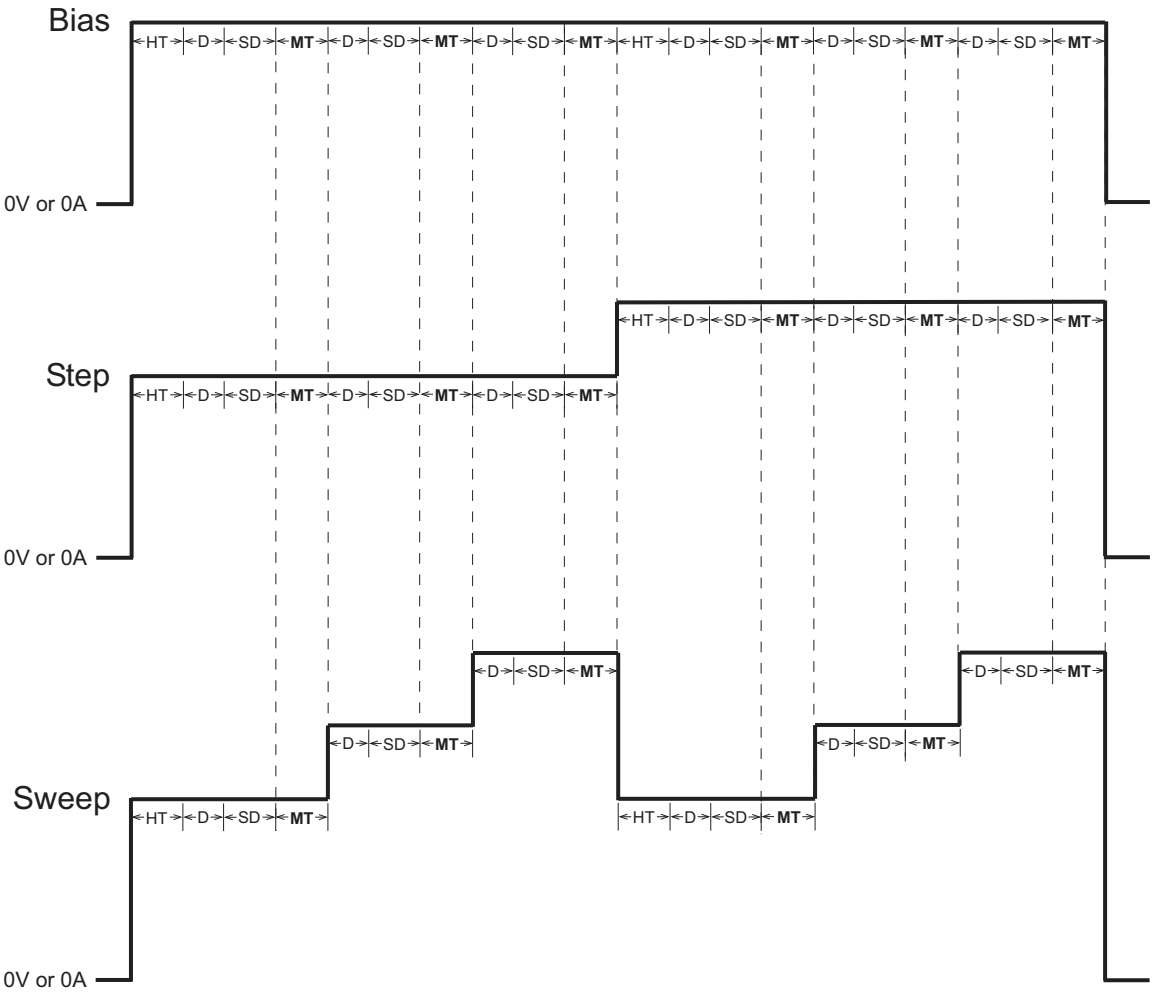
Sweep Delay setting: If you are using a sweep forcing function and desire some extra settling time before each measurement, you can specify an additional delay in the **Sweep Delay** edit box **(t**he time specified in the **Sweep Delay** edit box is independent of the delay discussed previously under Delay Factor setting later in this section). You can specify a **Sweep Delay** from 0 to 1000 s, in units of microseconds, milliseconds, and seconds.The default **Sweep Delay** is 0 s.

Hold Time setting: The starting voltage(s)/current(s) of a sweep may be substantially larger than the voltage/current increments of the sweep. Accordingly, the source settling time required to reach the starting voltage(s)/current(s) of a sweep may be substantially larger than the settling times required to increment the sweep. To compensate, you can specify a **Hold Time** delay to be applied only at the beginning of each sweep. You can specify a **Hold Time** delay of 0 to 1000 s, in units of microseconds, milliseconds, and seconds. The default **Hold Time** is 0 s.

Sweeping Mode timing diagram

Figure 6-153 shows source-measure timing for a test system using three SMUs. It shows basic timing between the three force functions: sweep, step, and bias.

Figure 6-153
Sweeping Mode timing diagram



HT = Hold Time
D = Delay (default delay x delay factor)
SD = Sweep Delay
MT = Measure Time

The timing elements in Figure 6-153 act as follows:

- Hold Time (HT): The sweep graph shows two sweeps that correspond to the two steps shown directly above in the step graph. Note that at the start of each sweep there is a hold time. The hold time is a global setting. Therefore, it is the same for all SMUs in the test system.
- Delay (D): The delay time, allows the source to settle, and is measurement-range dependent. All SMUs in the test system are synchronized. Therefore, the delay time applied by the most-delayed SMU is the delay time applied by all SMUs.
- Sweep Delay (SD): The sweep delay provides additional settling time for each step in the sweep. It is a global setting. and therefore is applied identically to all SMUs in the test system.
- Measure Time (MT): The measure time is determined by the **Filter Factor** and the **A/D Integration Time**. All SMUs in the test system are synchronized. Therefore, the Measure

Time (MT) for the SMU requiring the longest measure time is the same for all SMUs in the test system.

Understanding and configuring the Sampling Mode area of the Timing window

The sampling mode allows measuring of voltages/currents as a function of time while forcing constant voltages/currents. For example, sampling mode would be used to profile capacitor charging voltage while forcing a constant current. Time is measured relative to when the SMU(s) apply the forced voltage or current (that is, t = 0 at the step change from 0.0 V/0.0A to the applied voltage/current).

KITE enables the **Sampling Mode** option only when all terminals of the device under test are configured for static forcing functions: **Open**, **Common**, **Voltage Bias,** or **Current Bias** (refer back to Static forcing functions). If any terminal of the device under test is configured for a step or sweep forcing function, the **Sampling Mode** option is unavailable.

If an ITM is configured for the **Sampling Mode**, you can configure three **Sampling Mode** settings: **Interval, #Samples,** and **Hold Time**. You can configure these settings for all measurement-**Speed** modes: **Fast**, **Normal, Quiet,** and **Custom,** as discussed below.

Sampling Mode settings

The **Interval**, **#Samples** and **Hold Time** settings control the **Sampling Mode**, as follows:

Interval: Specifies the time between measurements (data points). **Interval** can be set from 0 to 1000 sec, in units of microseconds, milliseconds, and seconds. The default **Hold Time** is 0.1s.

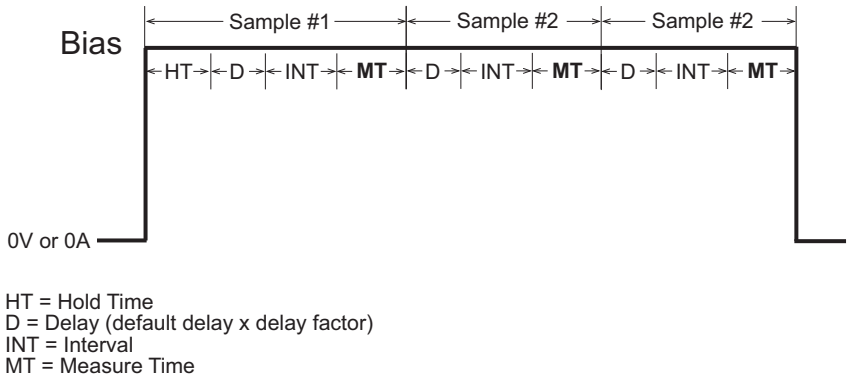
#Samples: Specifies the number of data points to be acquired. **#Samples** can be set from 1 to 4096. The default **#Samples** is 100**.**

Hold Time setting: After initial application of voltage/current by the SMU(s), the source settling time(s) can be substantial. To allow for settling, you can specify an extra **Hold Time** delay to be applied before making the first measurement. You can specify a **Hold Time** from 0 to 1000 s, in units of microseconds, milliseconds, and seconds. The default **Hold Time** is 1s.

Sampling Mode timing diagram

Figure 6-154 shows a timing diagram for the ***Sampling Mode*** test mode.

Figure 6-154
Sampling Mode timing diagram



HT = Hold Time
D = Delay (default delay x delay factor)
INT = Interval
MT = Measure Time

The timing elements in Figure 6-154 act as follows:

- If needed, a **Hold Time** (HT) can be used to allow for extra source settling after initial application of voltage(s)/current(s) by the SMU(s). **Hold Time** is a global setting and is therefore the same for all SMUs in the test system.
- A range-dependent delay (D) is automatically applied by an SMU before each measurement, to allow for source settling (refer to Delay Factor setting). All SMUs in the test system are synchronized. Therefore, the delay time applied by the most-delayed SMU is the delay time applied by all SMUs.

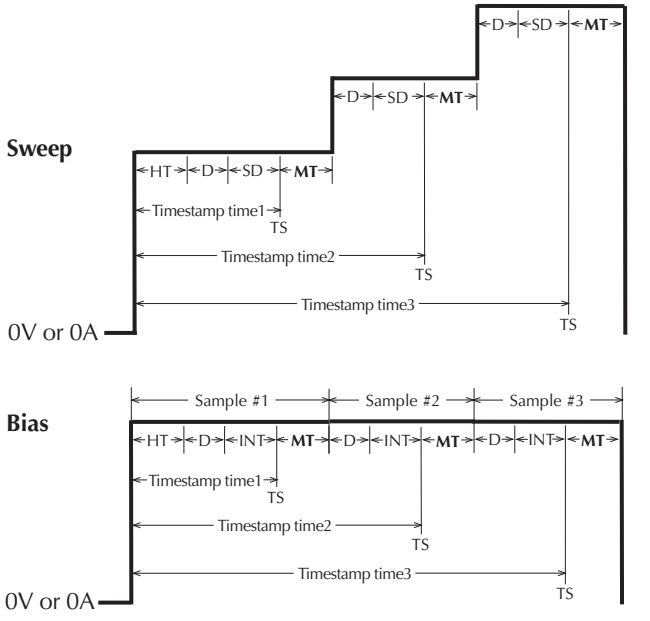
NOTE *In sampling mode, all device terminals are configured for static forcing functions:* ***Open****,* ***Common****,* ***Voltage Bias,*** *or* ***Current Bias****. Therefore, in sampling mode, the range-dependent delay may not be needed, because source settling time is not needed after the initial application of current or voltage. The delay can be set to 0 s by setting the* ***Delay Factor*** *to 0.*

- The Measure Time (MT) is determined by the **Filter Factor** and the **A/D Integration Time**. All SMUs in the test system are synchronized. Therefore, the Measure Time (MT) for the SMU requiring the longest measure time is the same for all SMUs in the test system.

Understanding and configuring the Timestamp Enabled checkbox

- When the **Timestamp Enabled** checkbox is checked, KITE records the elapsed time for each measurement. Each elapsed time value is placed in the **Sheet** tab **Data** worksheet in the same row as the measurement.
- Each elapsed time is measured relative to the beginning of the test, when voltage or current is first applied to the device. If KITE requires only one reading for a measurement, then KITE records the timestamp at the beginning of this reading. See Figure 6-155.

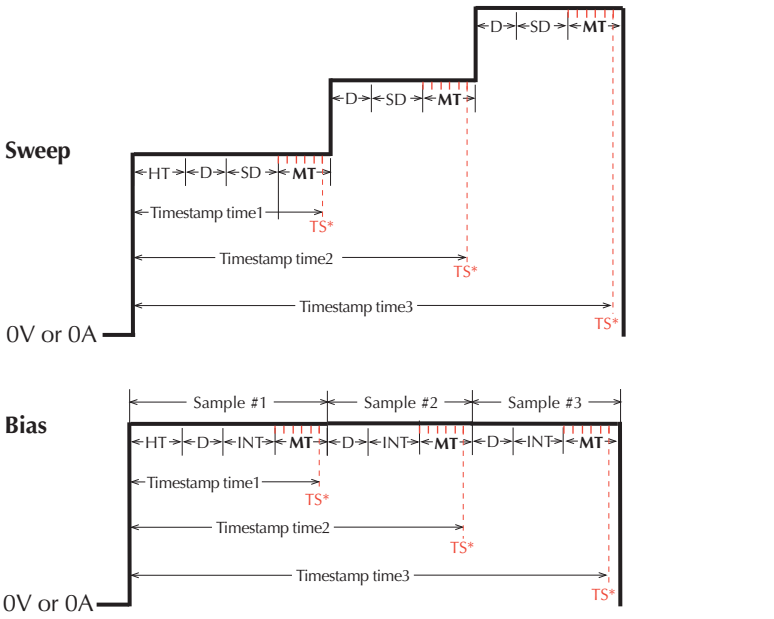
Figure 6-155
Application of timestamps when KITE requires only one reading for a measurement



TS = Timestamp at start of reading when one reading is taken
HT = Hold Time
D = Delay (default delay x delay factor)
SD = Sweep Delay
INT = Interval
MT = Measure Time

If KITE takes and averages multiple readings for a measurement, then KITE records the timestamp at the last of these readings. See Figure 6-156.

Figure 6-156
Application of timestamps when KITE requires multiple readings for a measurement



= Measurement consisting of multiple readings that are averaged
TS* = Timestamp at the beginning of the *last* reading of multiple readings
HT = Hold Time
D = Delay (default delay x delay factor)
SD = Sweep Delay
INT = Interval
MT = Measure Time

You can enable the timestamp for all measurement **Speed** modes: **Fast**, **Normal, Quiet,** and **Custom.**

Power On Sequence

When an ITM test is run, the SMUs for the given test power-on in a specific sequence. In the **ITM Timing** window (see Figure 6-152), the SMUs for the power-on sequence are identified by device terminals. For example, in Figure 6-152 the power-on sequence is **Source**, **Drain** and then **Gate**. The SMU connected to the source powers-on first. The SMU for the drain powers-on next, and the SMU for the gate powers-on last.

The power-on sequence can be changed by selecting (clicking) a terminal, and then using the **Move Up** or **Move Down** buttons to change its position in the sequence. SMU outputs can be disabled when the test is completed by selecting (checking) **Disable outputs at completion**.

WARNING ***If "Disable output at completion" is unchecked, the SMU outputs will remain at their last programmed levels when the ITM test is completed.***

If **Disable output at completion** is unchecked, for the next ITM or UTM in the sequence that uses the SMU, the SMU starts from it's previous state. If the next test does not use the SMU, then it remains in that state. If only a single ITM is selected to execute, check the **Reinitialized hardware when execution completes** setting in the **Tools > Options** dialog. If the setting is checked, then the SMU output will remain at the last programmed value for a brief time before being reinitialized. When KITE exits, all SMUs are placed in their default state.

Power On Delay: The first SMU in the sequence powers-on immediately when a test is run. Power-on delays can be set between all the SMUs in the test. The set delay for an SMU occurs after it powers-on, assuming there is another SMU in the power-on sequence.

For example, assume there are three SMUs in a test and the power-on sequence is SMU1, SMU2 and SMU3. Also assume that the power-on delay for SMU1 is set to 50 ms and the delay for SMU2 is set for 100 ms. When the test is started, the power-on sequence for the SMUs will be as follows:

SMU1 powers-on > 50 ms delay > SMU2 powers-on > 100 ms delay > SMU3 powers on

The **Power On Delay** field in the **Forcing Functions / Measure Options** window is used to set this delay, which can be set from 0 to 0.100 s.

FAQs (frequently asked questions) about the Timing window

- **Question**: Can I override the delays and filtering that are pre-programmed into the system?
  Answer: Yes. When the **Filter Factor** and **Delay Factor** are set to zero, the internal, pre-programmed values are ignored. The user can then specify a fixed **A/D Integration Time** of 0.01 to 10 PLC. Also, note that the following delays can be added:
  - A **Hold Time** delay of 0 to 999 s can be specified in the **Sweeping Mode** and **Sampling Mode** areas of the Timing window.
  - A **Sweep Delay** of 0 to 999 s can be specified in the **Sweeping Mode** area of the Timing window.
  - An **Interval** of 0 to 999 s can be specified in the **Sampling Mode** area of the Timing window, to space measurements as a function of time.
- **Question**: What are the best settings if the system requires long cables or includes a switch matrix?
  Answer: In general, cables and matrices increase the settling time. Therefore, the **Delay Factor** can be increased to allow for the added settling time. A little trial-and-error experimentation is needed. However, for a good quality switch, such as the Keithley Instruments Model 7174A Ultra Low Current matrix, you should not need to increase the **Delay Factor** by more than 2X.

Configuring Formulator calculations

The **Formulator**, accessible from an ITM **Definition** tab, allows you to perform simple in-test calculations on ITM data and complex post-test data calculations. The following operators and functions may be used for in-test, real-time calculations on ITM data:

- Operators: +, -, *, /, ^
- Functions: ABS, DELTA, DIFF, EXP, INTEG, LN, LOG, SQRT

A variety of other functions may be used for post-test calculations.

For details on using the **Formulator**, refer to Configuring Formulator calculations later in this section.

Saving the ITM configuration

Save the ITM configuration, by either of the following methods:

- Click the diskette icon at the top of the KITE window (shown below).

Figure 6-157
Diskette icon

- In the **File** menu, select **Save**.

ITM compliance exit conditions

Compliance limits are used to protect DUT from damage. When sourcing voltage, a current compliance limit can be set. When sourcing current, a voltage compliance limit can be set. If the compliance limit is reached, the voltage or current clamps at that level.

By default, a reached compliance limit does not affect the testing process. That is, the test or plan continues even though the source is in compliance. However, the test or plan can be set to exit (abort) if the source goes into compliance. The following exit conditions are available for the compliance condition:

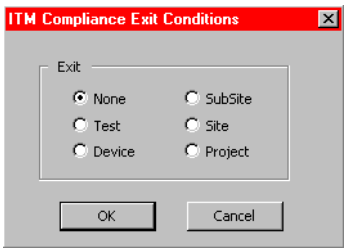
- **None**: The test or plan is not affected by the source going into compliance. This is the default setting.
- **Test**: The Model 4200-SCS exits the test presently being run. If running a plan, operation continues on to the next test.
- **Device**: The Model 4200-SCS exits the Device Plan presently being run. If configured to run another plan, operation continues on to that plan.
- **Subsite**: The Model 4200-SCS exits the Subsite Plan presently being run. If configured to run another Subsite Plan, operation continues on to that plan.
- **Site**: The Model 4200-SCS exits the Site. If configured to test another Site, operation continues on to that site.
- **Project**: The Model 4200-SCS exits the Project.

Setting the ITM compliance exit condition

The compliance exit conditions are accessible from the ITM **Definition** tab:

1. On the ITM **Definition** tab, click the **Exit Conditions** button.
2. From the **Exit Conditions** window (see Figure 6-158), select the desired exit condition and click **OK**.

Figure 6-158
ITM Compliance Exit Conditions



ITM Output Values

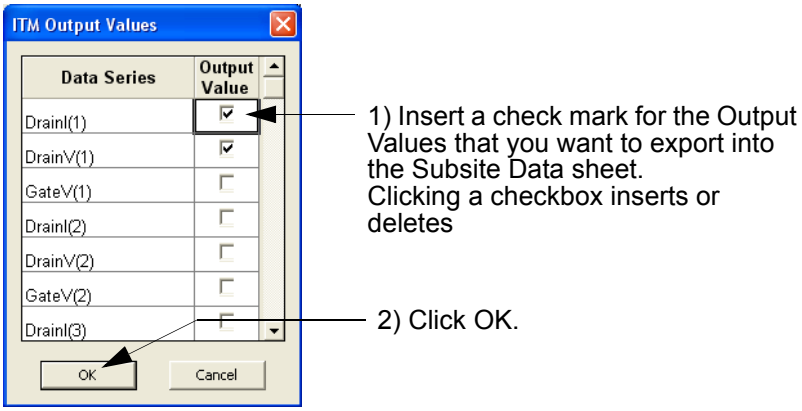
The measured readings for an ITM test can export (output) to a Subsite Data sheet for subsite cycling. The exported readings for an ITM test are called Output Values.

Each time a subsite is cycled, the measurements for the enabled output values are placed in the Subsite Data sheet. If for example, the subsite is cycled five times, there will be five measured readings (Output Values) for the ITM test. See Subsite cycling later in this section for details on subsite cycling.

Exporting Output Values: In the Project Navigator, double-click the desired ITM to open it. On the **Definition** tab, click the **Output Values** button (see Figure 6-4). The **ITM Output Values** window is shown in Figure 6-159. The two steps to export Output Values are shown in the drawing. Note that an ITM can have up to 20 Output Values.

NOTE *The measured readings for UTMs can also be exported to a Subsite Data sheet. See UTM Output Values.*

Figure 6-159
Exporting ITM Output Values to the Subsite Data sheet



Configuring the UTMs

After inserting ITMs and UTMs into your Project Plan, they must be configured to meet your test requirements. This subsection describes connection of Project Plan UTMs to user modules and user libraries and entering/editing of module parameters. ITM configuration was discussed previously under Configuring the Project Plan ITMs.

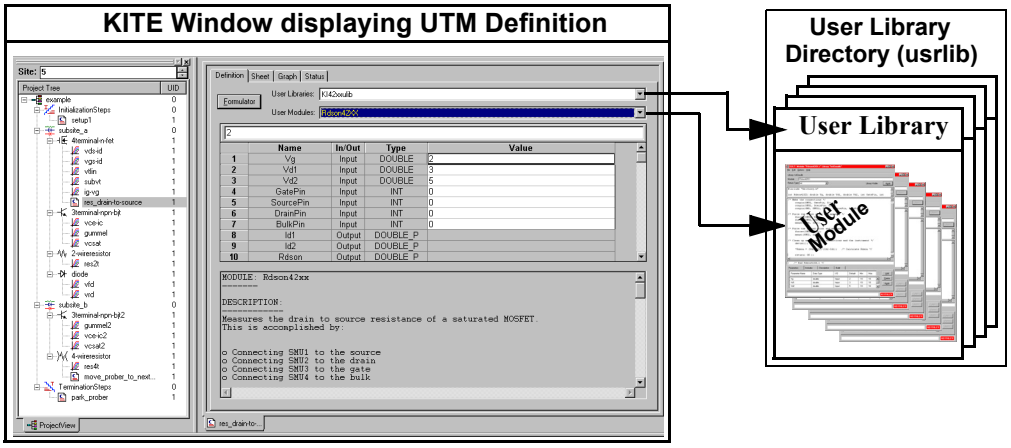
A Project Plan UTM is configured using the following main steps:

1. Open a UTM window for the Project Plan UTM.
2. Connect/reconnect the name of the UTM to an existing KULT created user module through the UTM **Definition** tab of a UTM window.
3. In the UTM **Definition** tab, enter the required parameters.
4. Configure, or change, **Formulator** calculations, if any.
5. Save the configuration.

These steps are illustrated graphically in Figure 6-160 and are discussed in detail in these next five subsections:

- Opening a UTM window
- Connecting/reconnecting the UTM to a user library and module
- Inputting the UTM parameters
- Configuring Formulator calculations
- Saving the UTM configuration
- UTM Output Values

Figure 6-160
Relationships between a Project Plan, a UTM, a user module, and user libraries



CAUTION **If the Project Plan contains multiple same-named instances of the UTM to be configured/reconfigured, ensure that you understand the shared and unique characteristics of same-named UTMs before configuring/ reconfiguring the UTM. Refer to Table 6-11 below.**

NOTE *The default user library directory can be controlled by KCON. Refer to "Keithley CONfiguration Utility (KCON)" in Section 7 for details.*

Table 6-11
Shared characteristics and unique characteristics for same named Project Plan UTMs

Characteristics that are shared between all instances of a Project Plan UTM having the same name. A change of one instance changes the definition of *all* instances identically.*	Characteristics that are unique to each instance of a Project Plan UTM having the same name. A change of one instance has no effect on any other instance.
Everything in the **Definition** tab for the UTM, except for the parameter settings. The shared characteristics include the following: • The specified user library • The specified user module • All **Formulator** formulas* • Output Values	• The parameter settings • Test data • Formulas in the **Calc** worksheet of the **Sheet** tab • Plot settings in the **Graph** tab • The Unique ID number (UID)

* Changing the user module or the user library causes all **Formulator** formulas to be deleted.

Opening a UTM window

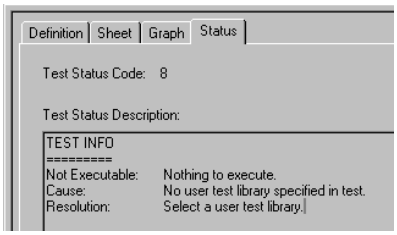
A UTM window allows you to enter information that defines a given UTM. A UTM window also allows you to view and analyze test data created by the UTM, both numerically and graphically.

- The **Definition** tab of the UTM window allows the user to specify a KULT created user library and user module and then to specify a limited number of test parameters. The included **Formulator** analysis tool allows you to perform calculations on test results and include the calculation results in the **Sheet** tab **Data** worksheet (see below).
- The **Sheet** tab displays the UTM results in its **Data** worksheet, in spreadsheet format. An independent spreadsheet in the **Sheet** tab, the **Calc** worksheet, allows the user to perform

custom, test-specific data analysis. A third worksheet, the **Settings** worksheet, displays the same settings information as in the **Definition** tab, but in spreadsheet format. Cells in the **Calc** worksheet may be hot-linked to cells in the **Data** and **Settings** worksheets.

- The **Graph** tab allows the user to create and export graphs of the test and test-analysis results. The **Graph** tab provides for flexible plot-data selection, formatting, annotation, and numerical coordinate display (using precision cursors).
- The **Status** tab monitors the current configuration status of the UTM, reporting its readiness for use and recommending additional preparations if necessary. A test-ready report for a UTM is the same as for an ITM (to view example **Status** tab reports for ITMs, refer to ITM Status tab later in this section). However, a test-not-ready report for a UTM is different than for an ITM. See the example in Figure 6-161.

Figure 6-161
Example Status tab report for an unconfigured UTM

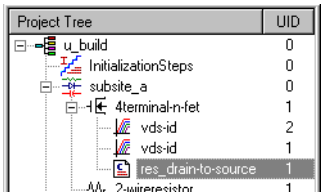


To configure, or reconfigure, a UTM, you need only the features of the **Definition** tab, which is the default tab of the UTM window.

Open a UTM window as follows:

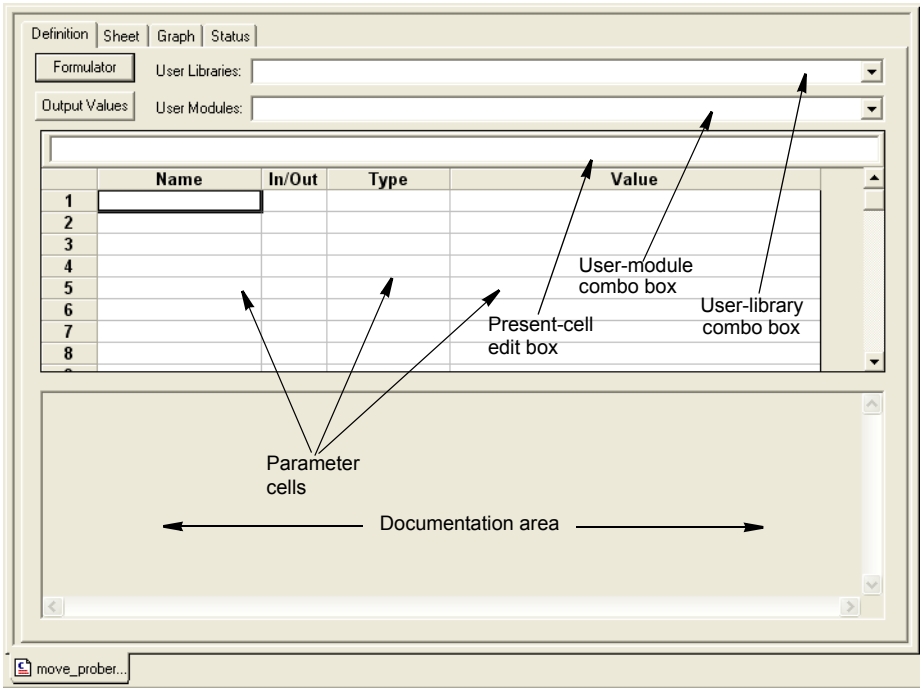
1. In the Project Navigator, locate the UTM that you wish to configure. For illustration, the **res_drain-to-source** UTM was selected from the **u_build** Project Plan (created under Building a completely new Project Plan). See Figure 6-162.

Figure 6-162
Opening a UTM Definition tab from the Project Navigator



2. Double-click on the UTM that you wish to configure (alternatively, if a Workspace window tab is already displayed for the UTM to be configured (see the bottom of the UTM window), the UTM is already open. Click the UTM Workspace window tab).
   - If this is a completely new UTM, inserted into the Project Plan only by name and not yet connected to a user module, a blank UTM **Definition** tab appears. See Figure 6-163, which was obtained by double-clicking the unconfigured **res_drain-to-source** UTM in the u_build Project Plan (Figure 6-162).

Figure 6-163
Blank UTM Definition tab



- If you previously configured the selected UTM or newly copied it into the Project Plan from a test library, a pre-configured UTM **Definition** tab appears. For example, if **res_drain-to-source** had been previously configured, it might look like Figure 6-164 or Figure 6-165.

A UTM **Definition** tab is divided as follows:

- **User Library combo box**: Displays the available user libraries and allows you to select one.
- **User Modules combo box**: Displays the user modules in the user library and allows you to select one.
- **Parameter input area**
  - **Parameter cells**: Spreadsheet-like cells that receive user inputs directly or display parameter titles/descriptions. As in a spreadsheet, each cell is designated by column and row (for example, the fourth column, second row is designated as D2). Any of the white cells next to parameter titles/descriptions are valid user-input cells.
  - **Present-cell edit box**: 1) Displays the contents of the presently-selected cell, and 2) provides an alternative, more spacious place to input or change the parameter of the currently selected user-input cell.
- **Documentation area**: Typically displays descriptions, examples, requirements, and so on. for the connected user module, whatever the module programmer deemed appropriate.
- **Formulator** button: Opens the **Formulator** tool, which, for UTMs that generate data, allows performance of simple in-test or complex post-test mathematical data analysis, such as parameter extractions.
- **Output Values** button: Use to specify (√) Output Values to export into the Subsite Data sheet.

Continue with Connecting/reconnecting the UTM to a user library and module.

Connecting/reconnecting the UTM to a user library and module

For a UTM to perform a task, it must be connected to a KULT created user library and user module. This subsection describes how to make a user-library / module connection for a new UTM or revise the user library/module for a previously configured UTM.

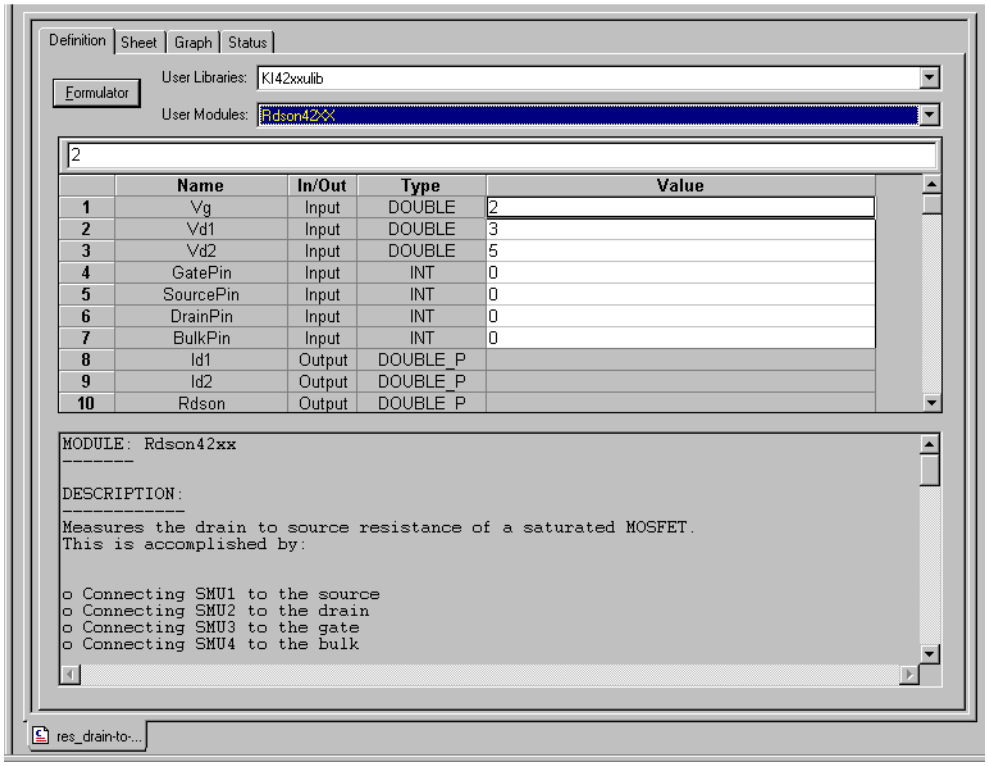
NOTE *If you only wish to change the parameters of a previously configured UTM, then skip to the next subsection, Inputting the UTM parameters.*

To select a user library and user module for the UTM, do the following:

1. Open the **User Library** scroll list, using the arrow key.
2. Select the appropriate user library.
3. Open the **User Modules** scroll list, using the arrow key.
4. Select the desired user module.

The UTM **Definition** tab now changes to reflect its connection to the new/revised user library and user module. For the **res_drain-to-source** UTM, the KI42xulib user library and the Rdson42XX module were selected. See Figure 6-164.

Figure 6-164
UTM Definition tab after selecting a user library and a user module



NOTE *If, using KULT, you modify the specified user module while the UTM **Definition** tab is open, the **Definition** tab does not visibly update to reflect the user module changes. You must first close the **Definition** tab and then reopen it, to see the changes.*

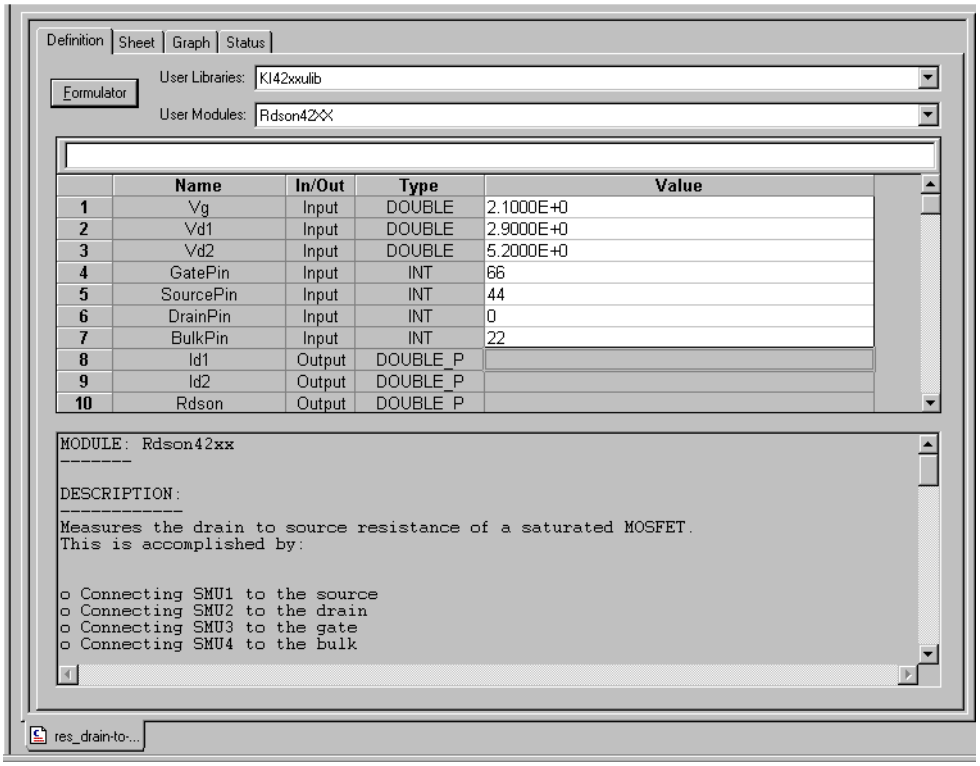
Inputting the UTM parameters

Most user modules provide default input parameter values (see the **Value** column in Figure 6-164). You may use the default value of any parameter or enter a new value, as follows:

1. In the parameter-cell matrix, double-click the cell in which you want to enter the new parameter value.
2. Enter the new parameter value either directly in the clicked cell or in the current cell edit box (located just below the **User Modules** combo box).

   Figure 6-165 illustrates revised parameter inputs for the **res_drain-to-source** UTM.

Figure 6-165
UTM Definition tab of Figure 6-164 after changing the parameter values



3. Continue with Configuring Formulator calculations.

Configuring Formulator calculations

The **Formulator**, accessible from the UTM **Definition** tab, allows you to perform simple and complex post-test data computations.

NOTE *Real-time **Formulator** calculations do not apply to UTM data. A UTM **Data** worksheet does not update until the test is finished.*

Continue with Saving the UTM configuration.

Saving the UTM configuration

Save the UTM configuration, by either of the following methods:

- Click the **Save** diskette icon at the top of the KITE window.
- In the **File** menu, select **Save**.

UTM Output Values

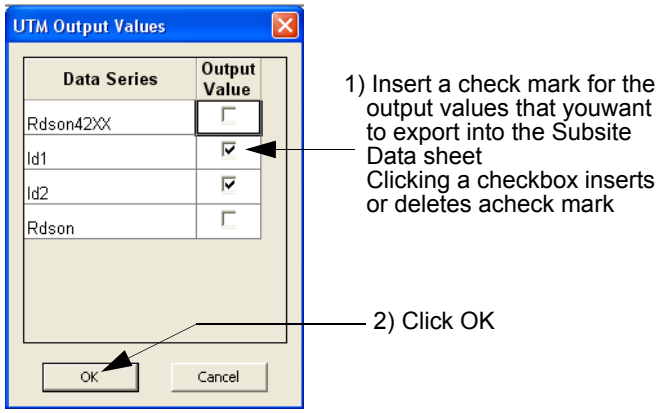
The measured readings for an UTM test can exported (output) to a Subsite Data sheet for subsite cycling. The exported readings for an UTM test are called Output Values.

Each time a subsite is cycled, the measurements for the enabled Output Values are placed in the Subsite Data sheet. If for example, the subsite is cycled five times, there will be five measured readings (Output Values) for the UTM test. For details, see Subsite cycling later in this section.

Exporting Output Values: In the Project Navigator, double-click the desired UTM to open it. On the **Definition** tab, click the **Output Values** button (see Figure 6-5). The UTM **Output Values** window is shown in Figure 6-166. The two steps to export Output Values are shown in the drawing. Note that a UTM can have up to 20 Output Values.

NOTE *The measured readings for ITMs can also be exported to a Subsite Data sheet. See ITM Output Values.*

Figure 6-166
Exporting UTM Output Values to the Subsite Data sheet



Submitting devices, ITMs, and UTMs to libraries

If you create a customized device or test and wish to reuse it in more than one place in a Project Plan or in other Project Plans, you must first submit it to a device or test library. The following two subsections show how:

- Submitting devices to a library
- Submitting tests to a library

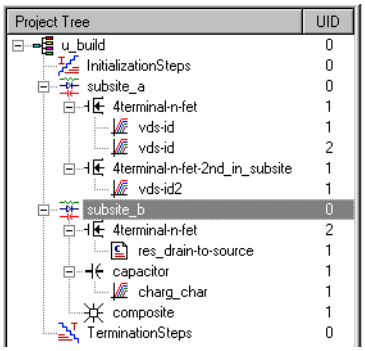
Submitting devices to a library

You may submit a Project Plan device (an empty Device Plan) to any device library, as long as you submit it under a name that does not duplicate a device name that is already in the library.

Submit a device as follows:

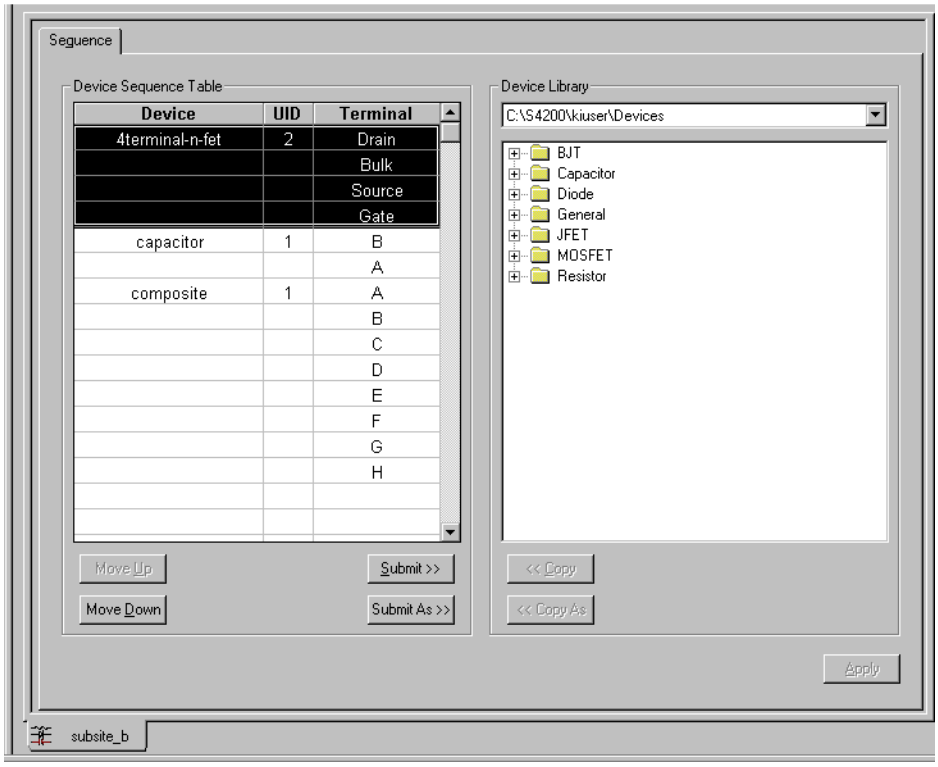
1. In the Project Navigator, locate the Subsite Plan that contains the Device Plan you wish to submit. Figure 6-167 highlights **subsite_b** of the **u_build** Project Plan (developed for illustration purposes during the course of Building a completely new Project Plan earlier in this section). The **subsite_b** plan presently contains an added **composite** Device Plan to be submitted.

Figure 6-167
Subsite Plan containing the Device Plan to be submitted



2. Double-click the Subsite Plan that contains the device(s) that you wish to submit. The Subsite Plan window opens. See Figure 6-168.

Figure 6-168
Subsite Plan window containing the Device Plan to be submitted



3. If you wish to submit the Device Plan to a device library directory other than the default device library directory,[6] select the alternate device library directory in the **Device Library** combo box of the Subsite Plan window.

NOTE *Only the default device-library directory is available in the **Device Library** combo box, unless other device library directories were previously added through the **Directories** tab of the KITE Options window. The KITE Options window is accessed through the **Tools > Options** menu.*

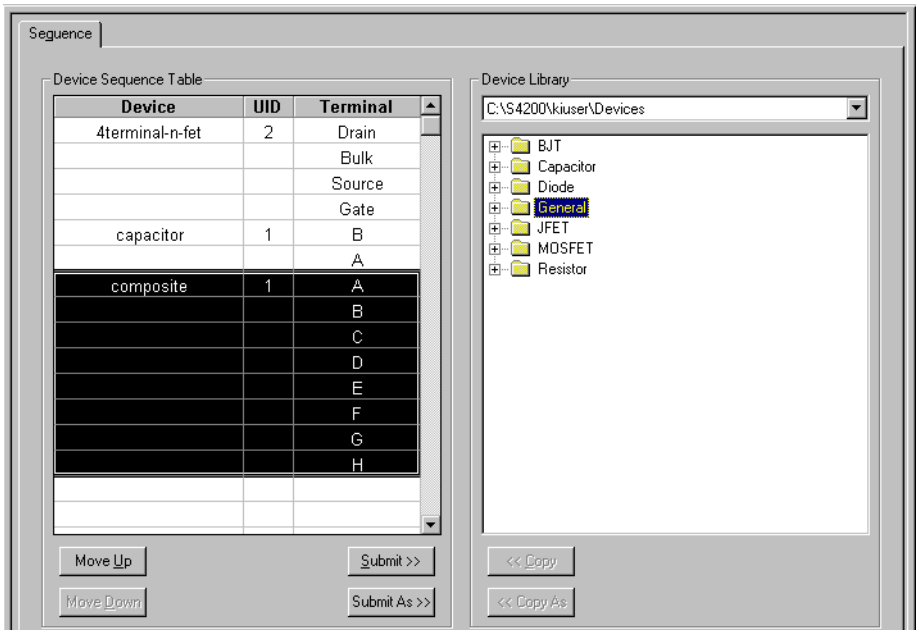
---

4. In the **Device Library** directory tree, select a destination folder that is appropriate for the device(s).
5. In the **Device Sequence Table** of the Subsite Plan window, select the devices(s) to be submitted.

NOTE *You may select and submit multiple Device Plans at the same time. To select a sequential group of Device Plans, hold down the **SHIFT** key while clicking on the first and last Device Plan in the sequence. To select a group of individual Device Plans, hold down the **CTRL** key while clicking on the individual Device Plans.*

Figure 6-169 shows 1) the **composite** device selected in the **Device Sequence Table,** and 2) the **General** destination folder selected in the **Device Library.**

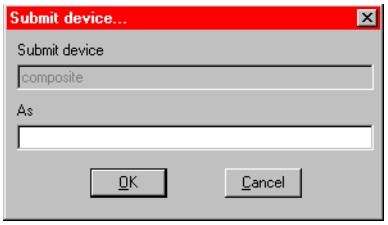
Figure 6-169
Selected device and destination folder



6. For example, the C:\S4200\kiuser\Devices factory-default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Devices.

6. Do one of the following:
   - If you wish to submit the selected device(s) with the original name(s), click the **Submit >>** button in the Subsite Plan window. The selected device(s) is submitted to the chosen folder.
   - Stop here. You have finished the device submission procedure.
   - If you wish to submit the selected device(s) under a different name(s), click the **Submit As >>** button in the Subsite Plan window. The **Submit device** dialog box opens, displaying the original name of the device (or, if you selected multiple devices, displaying the original name of one of the devices). See Figure 6-170.

Figure 6-170
Submit device dialog box



7. In the **As** edit box of the **Submit device** dialog box, type the submittal name for the device.
8. Click **OK**. One of the following occurs:
   - If you selected only one device in the **Device Sequence Table**, the selected device is submitted to the chosen folder under the new name. Stop here. You have finished the device submission procedure.
   - If you selected multiple devices in the **Device Sequence Table**, the following occurs:
     - The device that you renamed in step 7 is submitted to the chosen folder under the new name.
     - Then, another **Submit device** dialog box opens for another selected device.
9. Repeat steps 7 and 8 until all of the selected devices have been submitted (until no more **Submit device** dialog boxes open).

Submitting tests to a library

You may submit one or more ITMs or UTMs to any test library, however, you must submit them under names that do not duplicate test names that are already in the library.

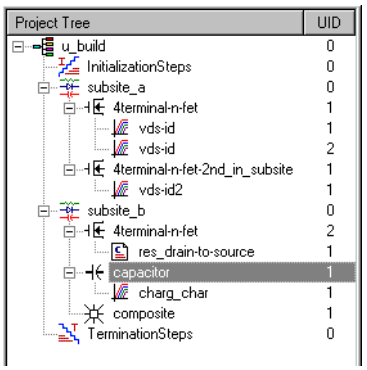
NOTE *Before submitting any UTM to a library, make sure that it is configured. If you try to submit an unconfigured UTM, KITE displays the message of Figure 6-171.*

Figure 6-171
Unconfigured UTM message

Submit the UTM(s) or ITM(s) (hereafter, mostly referred to simply as tests) as follows:

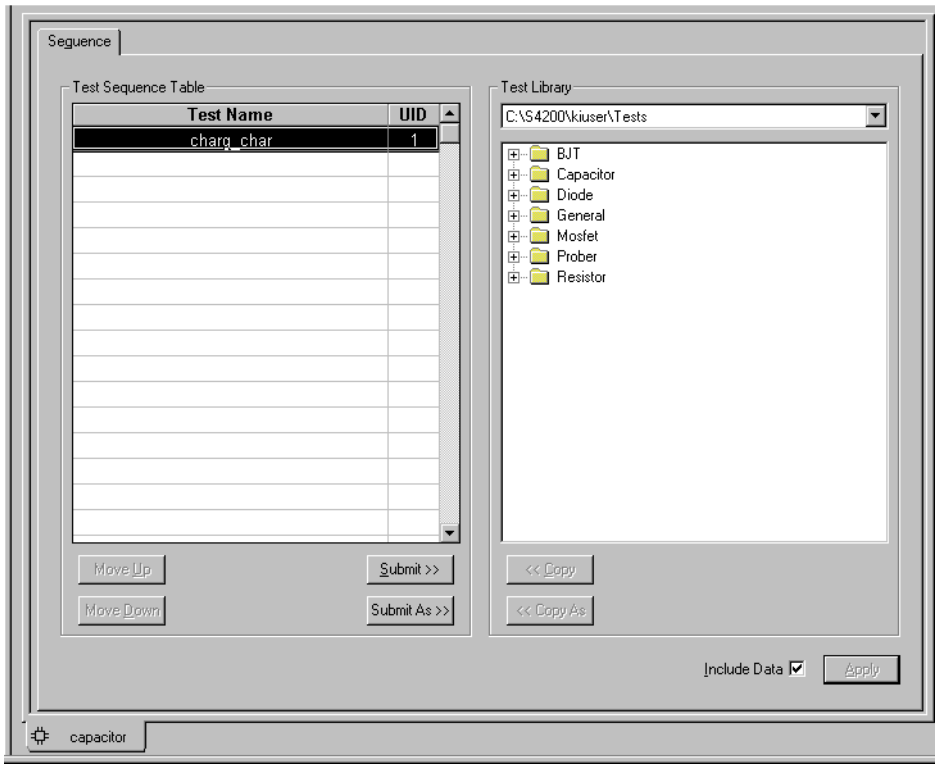
1. In the Project Navigator, locate the Device Plan that contains the test(s) that you wish to submit. Figure 6-172 highlights the **capacitor** Device Plan of the **u_build** Project Plan (developed for illustration purposes during the course of Building a completely new Project Plan earlier in this section). The **capacitor** Device Plan contains the **charg_char** ITM to be submitted.

Figure 6-172
Device Plan containing an ITM to be submitted



2. Double-click the Device Plan that contains the test(s) that you wish to submit. The Device Plan window opens. See Figure 6-173.

Figure 6-173
Device Plan window containing an ITM to be submitted



3. If you wish to submit the test(s) to a test library directory other than the default test library directory,[7] select the alternate test library directory in the **Test Library** combo box of the Device Plan window.

NOTE *Only the default test library directory is available in the **Test Library** combo box, unless other test library directories were previously added through the **Directories** tab of the KITE Options window. The KITE Options window is accessed through the **Tools > Options** menu.*

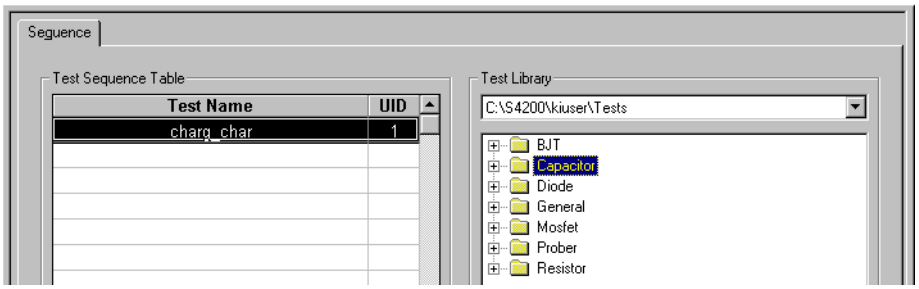
---

4. In the **Test Library** directory tree, select a destination folder that is appropriate for the test(s).
5. In the **Test Sequence Table** of the Device Plan window, select the test(s) to be submitted.

NOTE *You may select and submit multiple ITMs and UTMs at the same time. To select a sequential group of ITMs and UTMs, hold down the **SHIFT** key while clicking on the first and last ITM/UTM in the sequence. To select a group of individual ITMs and UTMs, hold down the **CTRL** key while clicking on the individual ITMs and UTMs.*

Figure 6-174 shows 1) the **charg_char** ITM selected in the **Test Sequence Table,** and 2) the **Capacitor** destination folder selected in the **Test Library.**

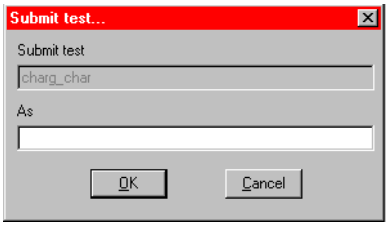
Figure 6-174
Selected ITM and destination folder



6. Do one of the following:
   - If you wish to submit the selected test(s) with the original name(s), click the **Submit >>** button in the Device Plan window. The selected test(s) is submitted to the chosen folder. Stop here. You have finished the test submission procedure.
   - If you wish to submit the selected test(s) under a different name(s), click the **Submit As >>** button in the Device Plan window. The **Submit test** dialog box opens, displaying the original name of the test (or, if you selected multiple tests, displaying the original name of one of the tests). See Figure 6-175.

7. For example, the C:\S4200\kiuser\Tests factory-default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Tests.

Figure 6-175
Submit test dialog box



7. In the **As** edit box of the **Submit test** dialog box, type the submittal name for the test.
8. Click **OK**. One of the following occurs:
   - If you selected only one test in the **Test Sequence Table**, the selected test is submitted to the chosen folder under the new name. Stop here. You have finished the test submission procedure.
   - If you selected multiple tests in the **Test Sequence Table**, the following occurs:
     - The test that you renamed in step 7 is submitted to the chosen folder under the new name.
     - Then, another **Submit test** dialog box opens for another selected test.
9. Repeat steps 7 and 8 until all of the selected tests have been submitted (until no more **Submit test** dialog boxes open).

Copying tests from a library

You can copy tests into or out of the KITE test library. Use the following procedure as a guide.

1. In the Project Navigator, locate the Device Plan of the test(s) to copy. Figure 6-176 shows the diode device selected.
2. Double-click the Device Plan that contains the test(s) that to submit. The Device Plan window opens (see Figure 6-177).
3. Highlight the test to copy into the chosen Device Plan. Figure 6-177 shows the Device Plan for the diode, with the pulse-diode test selected in the Test Library.
4. For the test, select or clear the Include Data check box (see Figure 6-177 for an example):
   - Clear the **Include Data** check box to allow test parameters to be copied, but the data sheets will be empty.
   - Select the **Include Data** check box to allow test parameters to be copied including the data sheets.
5. Do one of the following to copy the test (from Figure 6-177):
   a. To submit the selected test with the original name, click the **<< Copy** button in the Device Plan window. The selected test(s) is submitted to the chosen folder. If the test name already exists for the Device Plan, the Test Already Exists appears (Figure 6-178).
      - Clicking the **Copy As** button allows a different test name to be entered (in the same manner as described below in b.).
      - Clicking the **Copy and Add** button submits a duplicate test with a new UID. For a brief explanation of duplicate ITMs and UIDs, see Table 6-3.
   b. To submit the selected test under a different name, click the **<< Copy As** button in the Device Plan window. The Copy test dialog box opens displaying the original name of the test and a text box to accept the new test name (see Figure 6-179). Type in the name for the test and click the **OK** button. Figure 6-180 shows the Device plan after submitting the test.
6. Repeat steps 3, 4, and 5 for each test to be copied.

7. You can reorder the tests by selecting the test and clicking the **Move Up** and **Move Down** buttons.
8. When finished, click the **Apply** button (see Figure 6-180).

Figure 6-176
Project tree with diode device selected

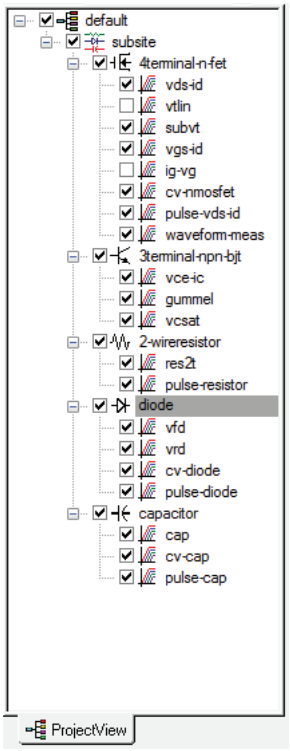


Figure 6-177
Diode device plan

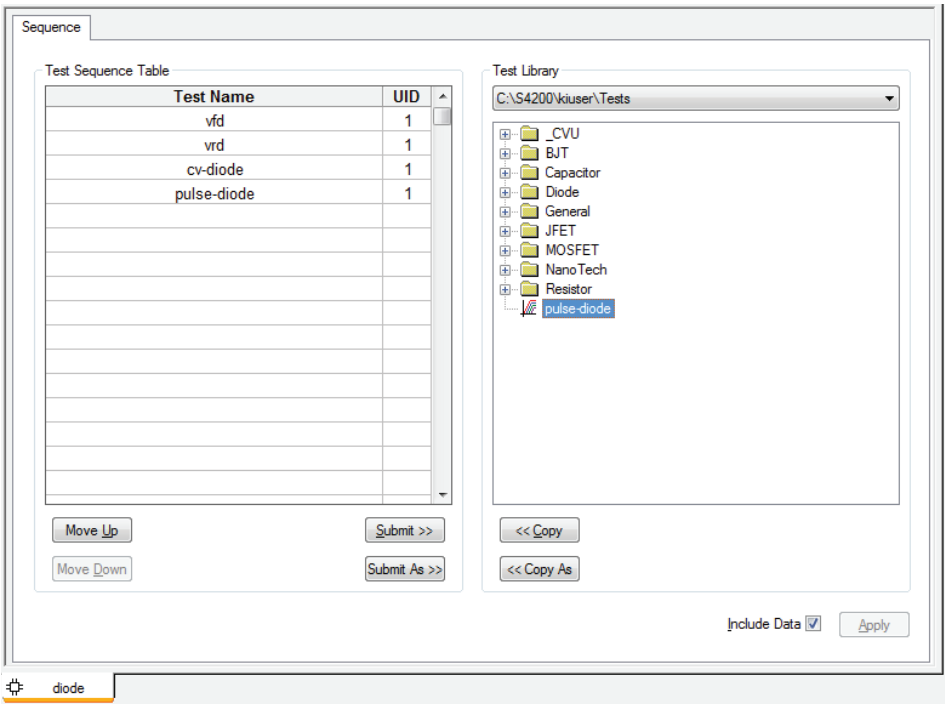


Figure 6-178
Test already exists prompt

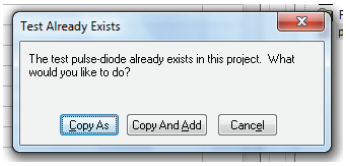


Figure 6-179
Copy as prompt

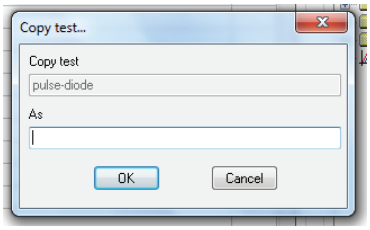
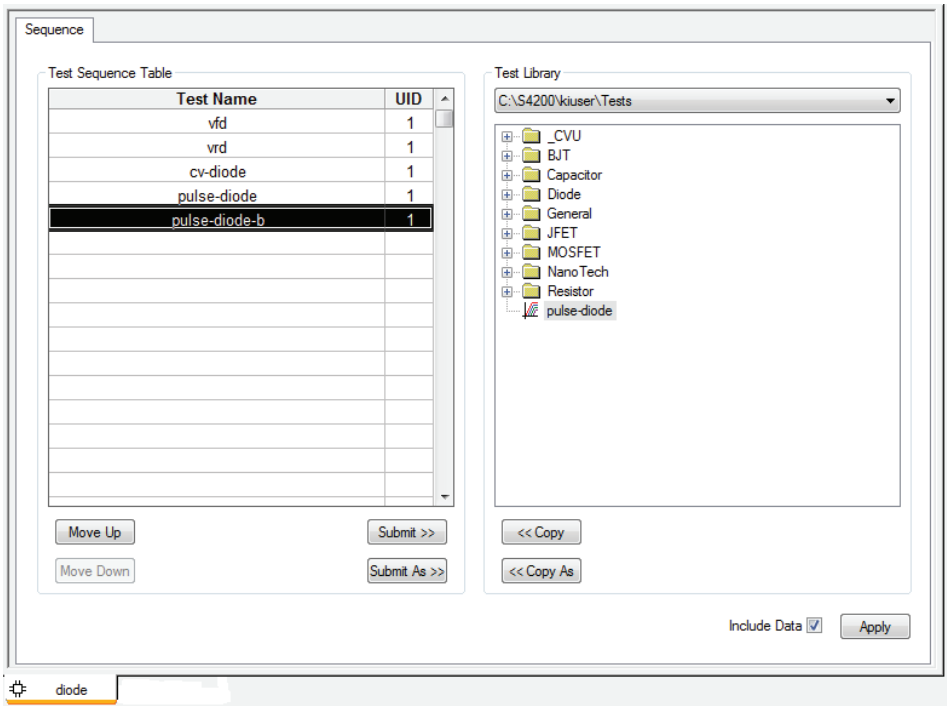


Figure 6-180
Device plan after submitting test



Executing Project Plans, Subsite Plans, Device Plans, and tests

You can execute an entire Project Plan or individual parts of the Project Plan. To describe how, this subsection discusses the following topics:

- Enabling tests (Project Navigator Checkboxes)
- 'Run' execution of Project Plans
- Run execution of individual tests and test sequences, which includes these topics:
  - 'Run' execution of individual Subsite Plans
  - 'Run' execution of individual Device Plans
  - 'Run' execution of individual tests

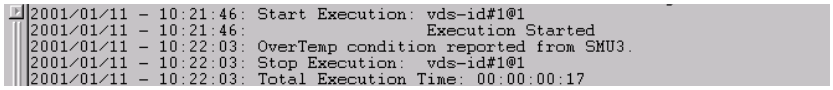
NOTE *If KITE detects an above-normal temperature condition at any SMU, it protects system outputs by preventing or aborting a run as follows:*

- ***New run attempt:*** *KITE prohibits execution of the run and reports the condition in the Message area of KITE window, as shown below:*

Figure 6-181
New run attempt error message

- *Existing run in progress: KITE aborts the run and reports the condition in the Message area of KITE window, similar to the following:*

Figure 6-182
KITE existing run in progress aborted



Enabling tests (Project Navigator Checkboxes)

Each component of the Project Plan has a checkbox. A check mark in a box indicates that the test or plan is enabled. The absence of a check mark indicates that the test or plan is disabled. Clicking a checkbox either inserts a check mark to enable or removes a check mark to disable. Only enabled (check marked) tests or plans can be run. See Project Navigator Checkboxes earlier in this section for details.

'Run' execution of Project Plans

Executing an entire Project Plan executes all of its components, initialization plans, Subsite Plans, Device Plans, ITMs, UTMs, and termination plans, in the order in which they appear in the Project Navigator:

1. Executes any initialization steps once, at the beginning.
2. Executes all Project Plan components (except for initialization and termination steps) multiple times until the specified number of sites are tested.
3. Executes any termination steps once, at the end.

When you execute a Project Plan, the data from each test is inserted into its own **Data** worksheet. Each new run updates the worksheet. For more about worksheets, refer to Understanding and using the Data worksheet of a Sheet tab.

NOTE *You must specify, in the Project window, the site numbers(s) with which collected data is to be labeled. You must also independently position the prober such that the site(s) to be evaluated on the wafer is identical to the site(s) that is specified in the Project window. This requirement applies whether you use a manual or semi-automatic prober and whether you evaluate one site or several.*

If you use a semi-automatic prober, understand that a KITE probe-step UTM only triggers movements that are already programmed in the prober controller. Each execution of the UTM advances the probe to the next site in this programmed sequence. Site numbers are not communicated between the prober and KITE, at this time. Therefore, if you evaluate multiple sites, the range of site numbers that you specify in the KITE Project window must agree with the sequence of site numbers in the prober controller program.

NOTE *Note that you can also generate appended worksheets for each test in a Project Plan, in addition to the Data worksheets. Refer to the separate discussion under "Append execution of tests, test sequences, and Project Plans."*

Execute a Project Plan as follows:

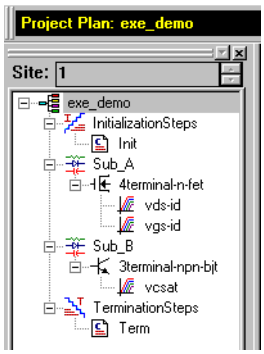
1. If the Project Plan to be executed is not yet open, open it as described under Opening an existing Project Plan.

If the Project Plan is already open but has not been saved, save the Project Plan by clicking the **Save All** icon at the top of the KITE screen (see below) or by clicking **Save All** in the KITE **File** menu.

Figure 6-183
KITE Save All icon

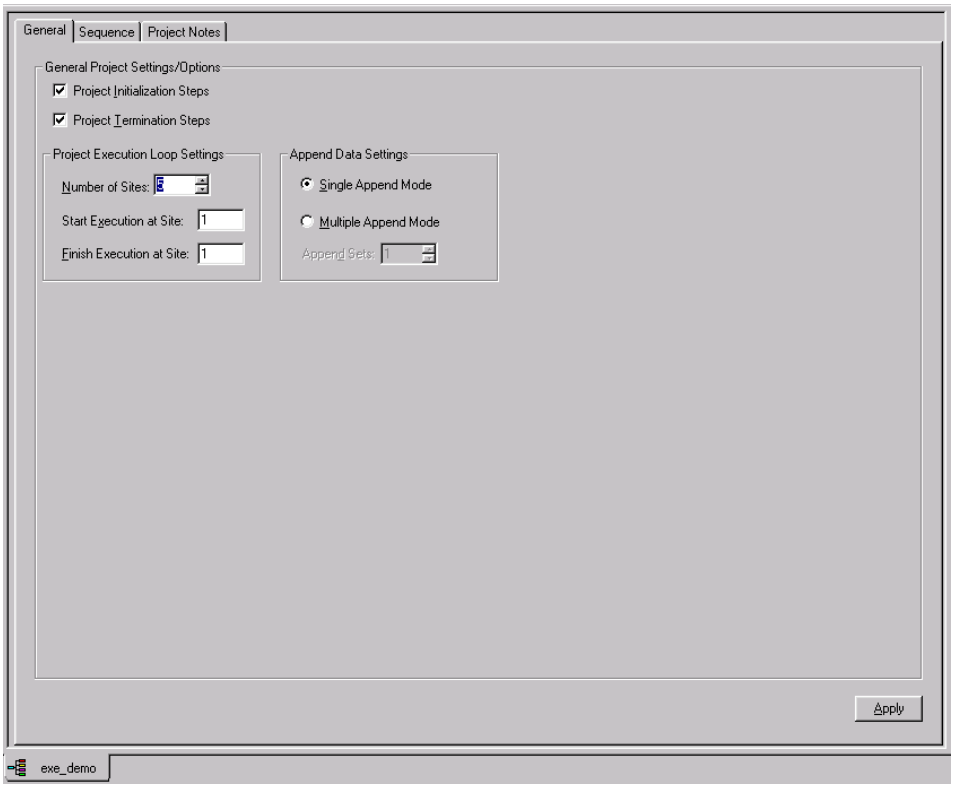
For illustration purposes, the **exe_demo** Project Plan was opened. See Figure 6-184.

Figure 6-184
exe_demo illustration Project Plan



2. Double-click on the Project Plan node (for example, on **exe_demo** per Figure 6-184). The Project window opens. Figure 6-185 shows an example Project window.

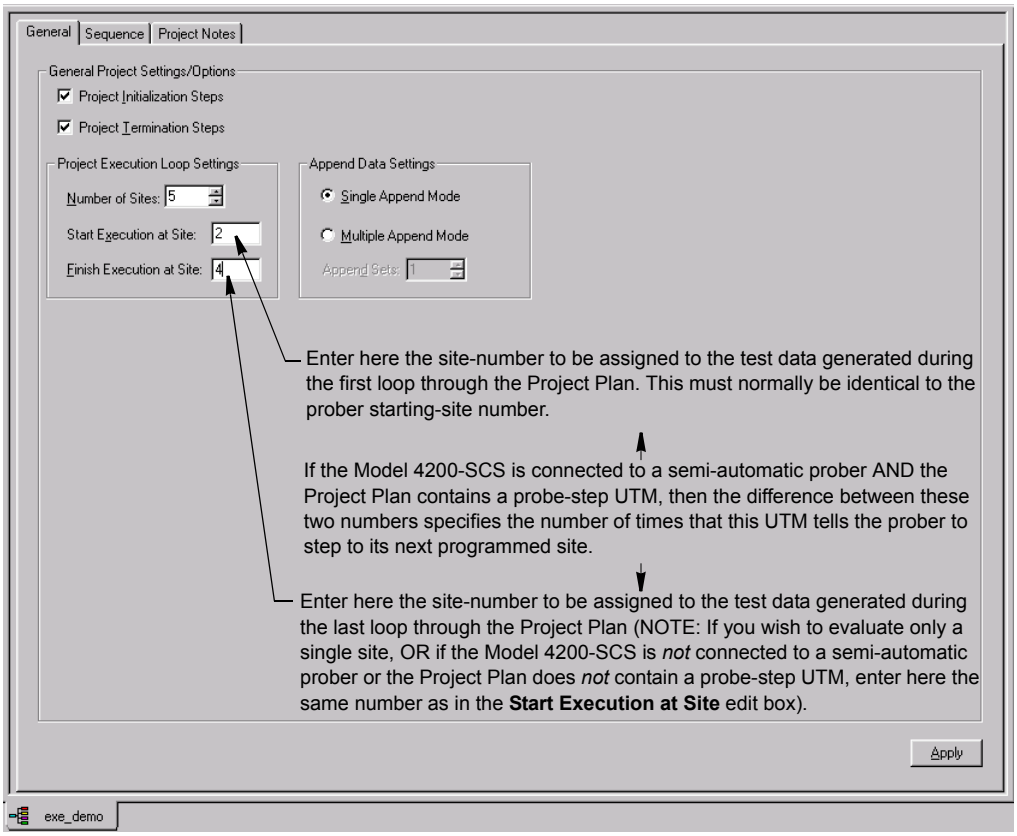
Figure 6-185
Example of Project window



3. In the Project window, set the **Start Execution at Site** and the **Finish Execution at Site** numbers as described in Figure 6-186.

NOTE *In the Project window, the **Number of Sites** is set 1) typically, to specify the number of sites that have been programmed in a prober controller; and 2) to automatically limit the **Finish Execution at Site** setting. Therefore, the **Finish Execution at Site** setting must be less than or equal to the **Number of Sites** setting.*

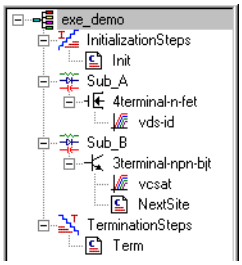
Figure 6-186
Project window site number settings



CAUTION **In the Project window, do not change the settings on the Project Initialization Steps or Project Termination Steps checkboxes. They are to be used only when building or modifying a Project Plan. Typically, UTMs under Initialization Steps and Termination Steps in the Project Plan perform such tasks as external instrument initialization and prober setup and parking. Unchecking Project Initialization Steps or Project Termination Steps checkboxes, and then clicking Apply, DELETES the UTMs. Checking these checkboxes does not add UTMs.**

4. Manually, or through the prober's controller, place the prober at the starting site.
5. At the top of the Project Navigator, select the **project** node. See the example in Figure 6-187.

Figure 6-187
Selecting the project node



NOTE *If you select a node other than the project node, KITE runs only the test or the test sequence at the node, and runs it only one time. Also, KITE labels the resulting data with the site number that is specified in the Site Navigator,* ***not*** *the* ***Start Execution at Site*** *number specified in the Project window. Refer to the next subsection, "Run execution of individual tests and test sequences."*

6. Start execution. Click the green triangular **Run Test/Plan** icon at the top of the KITE screen (see below), select **Run** in the KITE **Run** menu, or press the F6 keyboard key.

Figure 6-188
KITE Run Test/Plan icon

The green **Run Test/Plan** toolbar button becomes gray, the Project Plan executes, and the square **Abort Test/Plan** toolbar button illuminates red for the duration of the execution (see below).

Figure 6-189
KITE Abort Test/Plan button

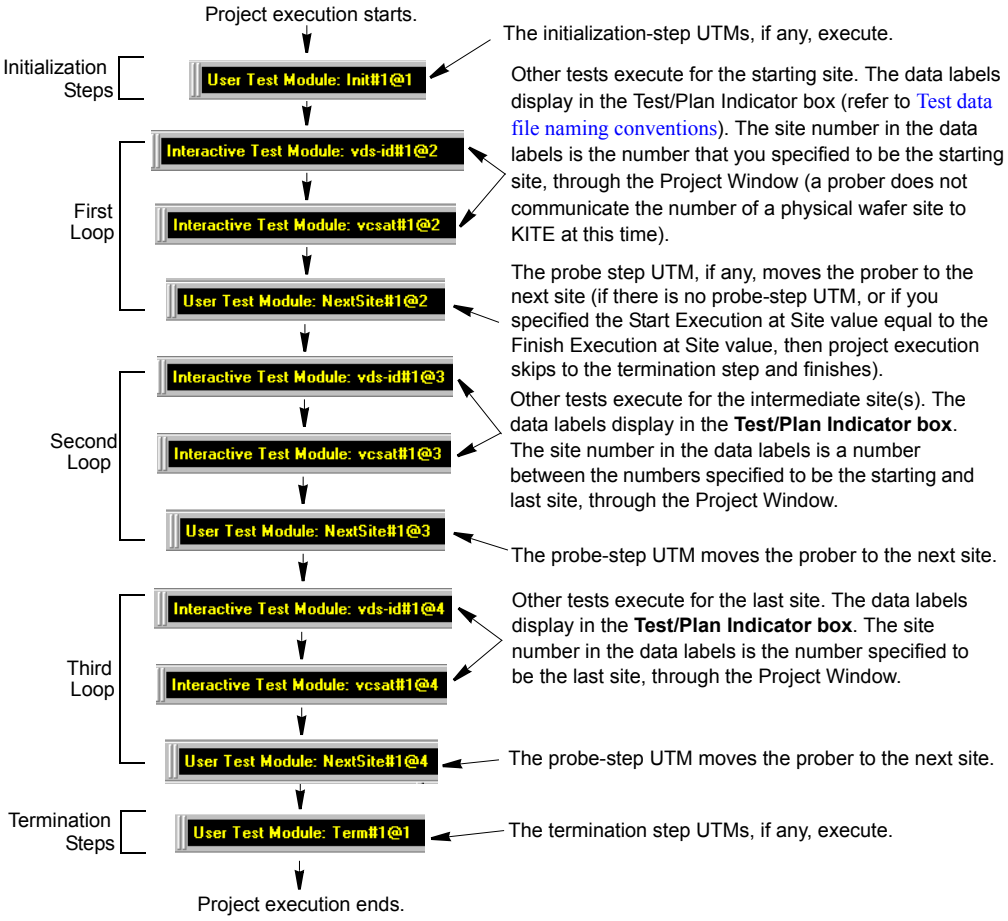
NOTE *If you should need to abort the subsite-plan execution, click the red* ***Abort Test/Plan*** *toolbar button or press the* ***PAUSE/BREAK*** *keyboard key.*

Simultaneously, the Execution Indicator toolbar button (see below) traces miniature curves and changes color for the duration of the project-plan execution.

Figure 6-190
KITE Execution Indicator button

Figure 6-191 illustrates the execution process for the **exe_demo** Project Plan, which was configured as shown in Figure 6-186.

Figure 6-191
Multi-site execution process, as displayed in the Test/Plan Indicator box



As each ITM and UTM in the Project Plan executes, the Test/Plan Indicator box displays its data label (refer to Test data file naming conventions). The data label identifies the site that is presently being evaluated, based on the numbers specified in the **Start Execution at Site** and **Finish Execution at Sit**e edit boxes.

Run execution of individual tests and test sequences

Executing individual tests and test sequences is somewhat different from executing a full Project Plan. The next three subsections discuss execution of individual Subsite Plans, Device Plans, and tests.

NOTE *Each time you execute a test or test sequence using the **Run** button, as described below, the data from each test is inserted into its own **Data** worksheet. Each new run updates this worksheet (for more about worksheets, refer to Understanding and using the Data worksheet of a Sheet tab). However, note that you can also generate appended worksheets for tests, test sequences. Refer to the separate discussion under Append execution of tests, test sequences, and Project Plans.*

'Run' execution of individual Subsite Plans

Executing a Subsite Plan executes only the components assigned to it (all of its Device Plans, ITMs, and UTMs) in the order in which they appear in the Project Navigator. No initialization steps, termination steps, or other Subsite Plans are executed.

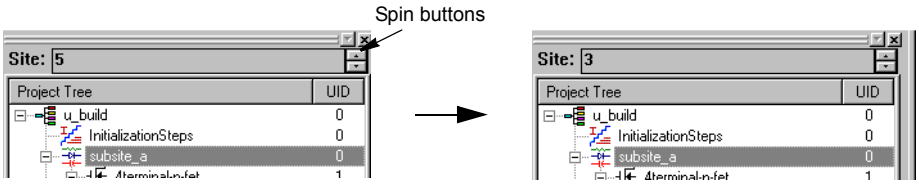
To execute a Subsite Plan, do the following:

1. If the Project Plan containing the Subsite Plan to be executed is not yet open, open the Project Plan as described under Opening an existing Project Plan.
   If the Project Plan is already open but has not been saved, save the Project Plan by clicking the **Save All** icon at the top of the KITE screen (see below) or by clicking **Save All** in the KITE **File** menu.

Figure 6-192
Save All icon

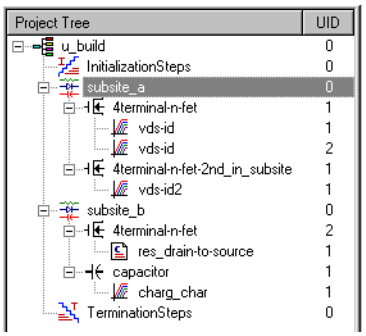
2. If the Model 4200-SCS is connected to a prober, do the following:
   a. Place the probe at the site that contains the subsite to be evaluated.
   b. Using the Site Navigator (located above the Project Navigator), scroll to the number of the site where you placed the probe in step 2a (use the spin buttons). See Figure 6-193.

Figure 6-193
Specifying the present probe-location site number through the Site Navigator



3. In the Project Navigator, select the name of the Subsite Plan to be executed. For example, to execute the **subsite_a** Subsite Plan of the **u_build** Project Plan (created under Building, modifying, and deleting a Project Plan earlier in this section), select **subsite_a**. See Figure 6-194.

Figure 6-194
Selecting a Subsite Plan for execution

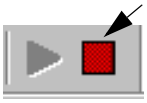


4. Start execution. Click the green triangular **Run Test/Plan** toolbar button (see below), select **Run** in the KITE **Run** menu, or press the **F6** keyboard key.

Figure 6-195
KITE Run Test/Plan icon

The green **Run Test/Plan** toolbar button becomes gray, the Subsite Plan executes, and the square **Abort Test/Plan** icon illuminates red for the duration of the test (see below).

Figure 6-196
KITE Abort Test/Plan button



NOTE *If you should need to abort the Subsite Plan execution, click the red **Abort Test/Plan** toolbar button or press the **PAUSE/BREAK** keyboard key.*

Simultaneously, the Execution Indicator (see below) traces miniature curves and changes color for the duration of the Subsite Plan execution.

Figure 6-197
KITE Execution Indicator button

As each test in the Subsite Plan executes, the Test/Plan Indicator box displays the data label for the test (refer to Test data file naming conventions later in this section). The data label includes the test identity and the site number that you specified in the Site Navigator. See the example below.

Figure 6-198
KITE Test/Plan indicator box

Subsite cycling

If a Subsite Plan is configured for subsite cycling, it will be repeated a specified number of times. For example, if the subsite plan is configured to cycle two times, the Subsite Plan will run two times. Test data is acquired for all cycles that are run. For details, see Subsite cycling later in this section.

Subsite cycling is started by clicking the following **Run Test/Plan and Cycle Subsites** button:

Figure 6-199
KITE Run Test/Plan and Cycle Subsites button

Subsite cycling can be terminated at any time by clicking the red Abort Test/Plan button.

'Run' execution of individual Device Plans

Executing a Device Plan executes only the components assigned to it—all of its ITMs and UTMs—in the order in which they appear in the Project Navigator. No initialization and termination steps and no other Subsite Plans or Device Plans are executed.

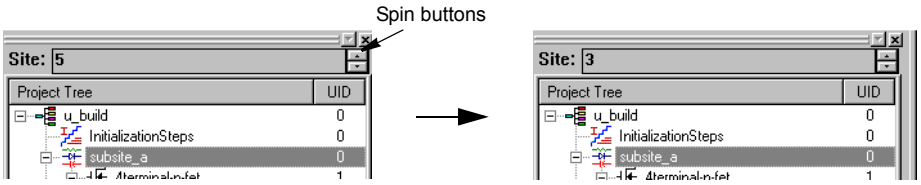
To execute a Device Plan, do the following:

1. If the Project Plan containing the Device Plan to be executed is not yet open, open the Project Plan as described under Opening an existing Project Plan later in this section.
   If the Project Plan is already open but has not been saved, save the Project Plan by clicking the **Save All** icon at the top of the KITE screen (see below) or by clicking **Save All** in the KITE **File** menu.

Figure 6-200
Save All icon

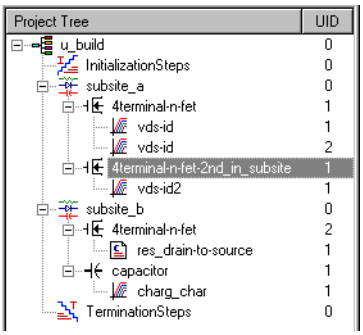
2. If the Model 4200-SCS is connected to a prober, do the following:
   a. Place the probe at the site that contains the device to be evaluated.
   b. Using the Site Navigator (located above the Project Navigator) scroll to the number of the site where you placed the probe in step 2a (Use the spin buttons). See Figure 6-201.

Figure 6-201
Specifying the present probe location site number through the Site Navigator



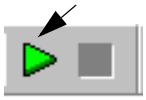
3. In the Project Navigator, select the name of the Device Plan to be executed. For example, consider the execution of a Device Plan in the **u_build** Project Plan (created under Building, modifying, and deleting a Project Plan). Figure 6-202 shows selection of the **4terminal-n-fet_2nd_in_subsite** Device Plan, which is located in **subsite_a**.

Figure 6-202
Selecting a Device Plan for execution



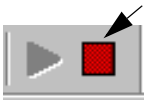
4. Start execution. Click the green triangular **Run Test/Plan** toolbar button, select **Run** in the KITE **Run** menu, or press the **F6** keyboard key.

Figure 6-203
KITE Run Test/Plan icon



The green **Run Test/Plan** icon becomes gray, the Device Plan executes, and the square **Abort Test/Plan** icon illuminates red for the duration of the test (see below).

Figure 6-204
KITE Abort Test/Plan button



NOTE *If you should need to abort the Device Plan execution, click the red **Abort Test/Plan** toolbar button or press the **PAUSE/BREAK** keyboard key.*

Simultaneously, the Execution Indicator toolbar button (see below) traces miniature curves and changes color for the duration of the Device Plan execution.

Figure 6-205
KITE Execution Indicator button

As each test in the Device Plan executes, the Test/Plan Indicator box displays the data label for the test (refer to Test data file naming conventions later in this section). The data label includes the test identity and the site number that you specified in the Site Navigator. See the example below.

Figure 6-206
Data label for test in progress

'Run' execution of individual tests

Executing an individual ITM or UTM plan executes only that specific test. No initialization or termination steps and no other Subsite Plans, Device Plans, or tests are executed.

To execute an ITM or UTM, do the following:

1. If the Project Plan containing the ITM or UTM to be executed is not yet open, open the Project Plan as described under Opening an existing Project Plan later in this section.

   If the Project Plan is already open but has not been saved, save the Project Plan by clicking the **Save All** icon at the top of the KITE screen (see below) or by clicking **Save All** in the KITE **File** menu.

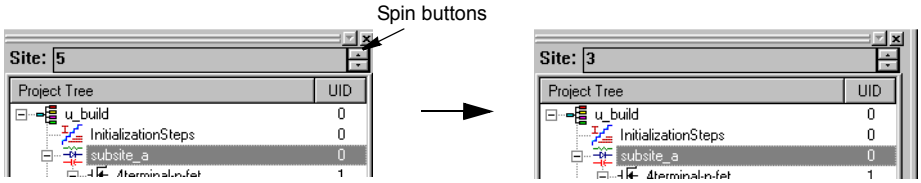
Figure 6-207
KITE Save All icon

2. If the Model 4200-SCS is connected to a prober, do the following:

a. Place the probe at the site that contains the ITM or UTM to be evaluated.

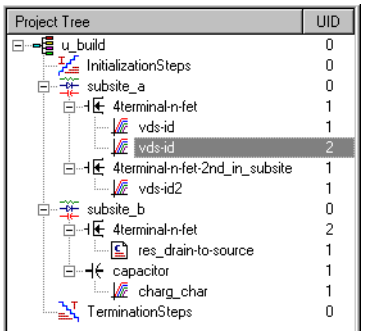
b. Using the Site Navigator (located above the Project Navigator), scroll to the number of the site where you placed the probe in step 2a (use the spin buttons). See Figure 6-208.

Figure 6-208
Specifying the probe-location site number through the Site Navigator



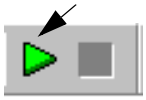
3. In the Project Navigator, select the name of the ITM or UTM to be executed. For example, consider the execution of an ITM in the **u_build** Project Plan (created under Building, modifying, and deleting a Project Plan). Figure 6-209 shows selection of the second **"vds-id"** ITM in the **4terminal-n-fet_2nd_in_subsite** Device Plan.

Figure 6-209
Selecting a test for execution



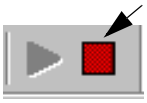
4. Start execution. Click the green triangular **Run Test/Plan** toolbar button, select **Run** in the KITE **Run** menu, or press the **F6** keyboard key.

Figure 6-210
KITE Run Test/Plan button



The green **Run Test/Plan** toolbar button becomes gray, the ITM or UTM executes, and the square **Abort Test/Plan** icon illuminates red for the duration of the test (see below).

Figure 6-211
KITE Test/Plan Abort button



NOTE *If you should need to abort the ITM or UTM execution, click the red **Abort Test/Plan** toolbar button or press the **PAUSE/BREAK** keyboard key.*

Simultaneously, the Execution Indicator toolbar button (see below) traces miniature curves and changes color for the duration of the test.

Figure 6-212
KITE Execution Indicator button

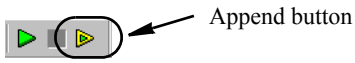
As the test executes, the Test/Plan Indicator box displays its data label (refer to Test data file naming conventions). The data label includes the test identity and the site number that you specified in the Site Navigator. See the example below.

Figure 6-213
Data label for test in progress

Append execution of tests, test sequences, and Project Plans

NOTE *There is a special execution button to initiate append executions. See the figure below:*

Figure 6-214
Append button



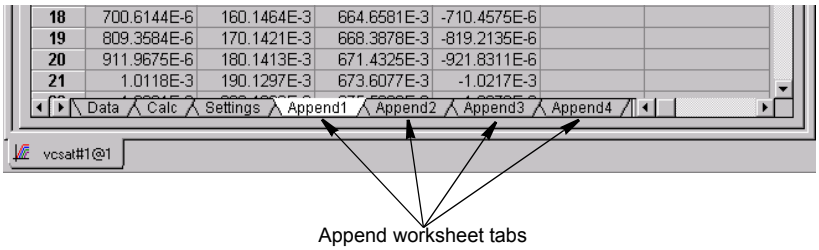
Understanding Append execution

With a **Run** execution, described above, only one **Data** worksheet is associated with each specific test (refer to Understanding and using the Data worksheet of a Sheet tab). This **Data** worksheet contains the data from the last run of the test; each new **Run** execution updates the worksheet. However, KITE also allows **Append** executions, which have the following characteristics:

- Each **Append** execution generates an additional **Append** worksheet (**Append1**, **Append2**,...and so on). for each additional run of the test.[8] KITE automatically stores the **Append** worksheets until a user-settable limit is reached (minimum = 1, maximum = 20). For each **Append** execution thereafter, the new **Append** data overwrites the most recent **Append** data. For example, if the user-settable Append limit is 4, the data for the fifth **Append** execution overwrites the data from the fourth **Append** execution. Likewise, the data for the sixth **Append** execution overwrites the data from the fifth Append execution, and so on.
- Each **Append** worksheet is part of the **Sheet** tab for the test (which is an Excel-compatible workbook). Each **Append** worksheet is displayed as one of the following tabs at the bottom of the **Sheet** tab: **Append1**, **Append2**, **Append3**... and so on. See Figure 6-215 and refer to Understanding and using Append worksheets of a Sheet tab.

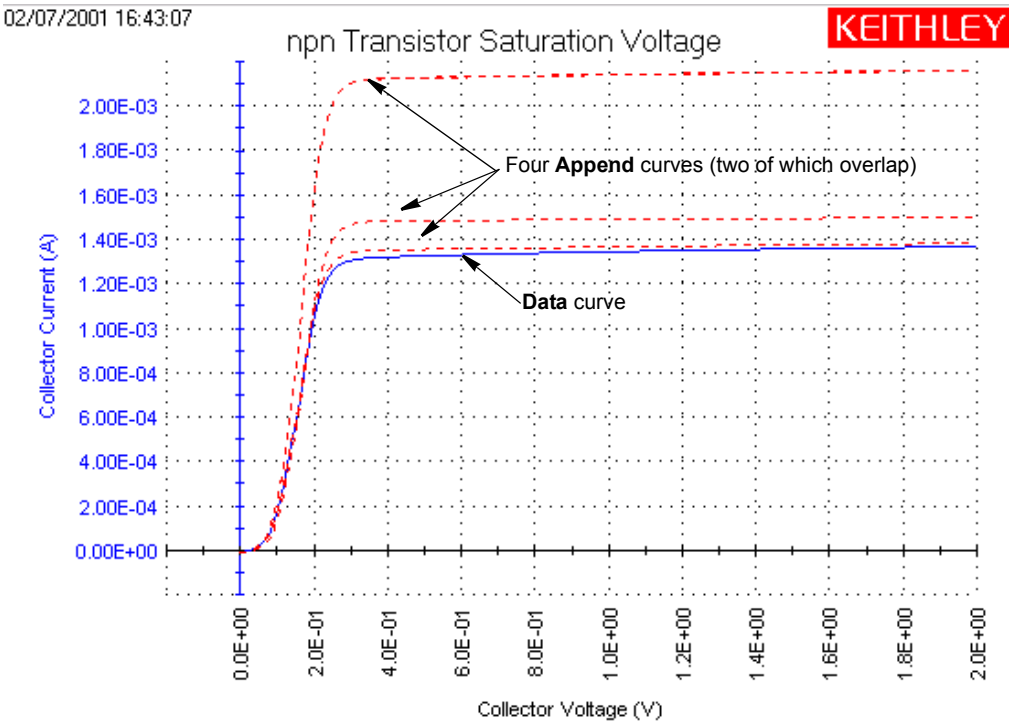
8. In addition to the **Data** worksheet generated by **Run** execution.

Figure 6-215
Append worksheet tab illustration



- In the **Graph** tab, the **Append** data curves for a test append to (layer on top of) the **Run**-data curves. Figure 6-216 shows the results of four **Append** mode runs.

Figure 6-216
Appended graph example



For details about graphing the appended data, refer to Appending curves from multiple runs on a single graph.

The **Append** mode may be applied to an entire test sequence (a Device Plan or Subsite Plan) as well as to a solitary test. Each time the sequence is run, the results of each test in the sequence are placed in the appropriate data sheet and appended to the appropriate graph.

NOTE *You cannot **Append** execute an entire Project Plan; you can Append execute only a test sequence or individual test.*

Specifying the maximum number of Append worksheets

Understanding the maximum number of Append worksheets

The maximum number of **Append** worksheets is set in the Project window. Therefore, the maximum number of **Append** worksheets applies to all tests in the entire Project Plan.

After the maximum number of **Append** worksheets have been generated, the data from each subsequent **Append** execution overwrites the most recent data (the data in the highest-numbered worksheet). For example, consider the following:

- If the maximum number of **Append** worksheets is 1 (the default setting), each new set of **Append** data overwrites the previous set.
- If the maximum number of **Append** worksheets is 4, the data for the fifth **Append** execution overwrites the data from the fourth **Append** execution. Likewise, the data for the sixth **Append** execution overwrites the data from the fifth **Append** execution, and so on.

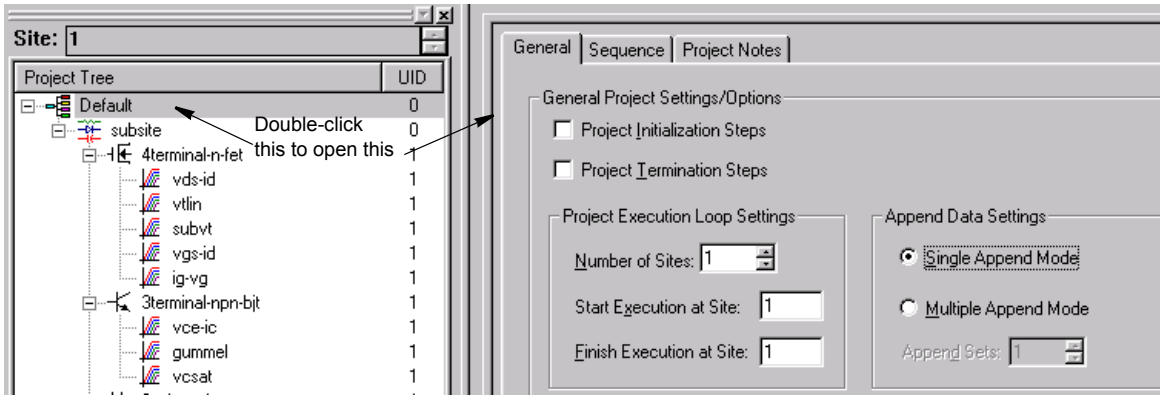
Setting the maximum number of Append worksheets

CAUTION **If you reduce the maximum number of Append worksheets, any existing higher-numbered worksheets are discarded for the entire Project Plan. For example, if you reduce the maximum number of Append worksheets to some new value "n," the following worksheets are discarded for each test in the entire Project Plan:**
Append(n+1), Append(n+2), ... Append(n+old_max_number).

Set the maximum number of Append executions in the Project window, as follows:

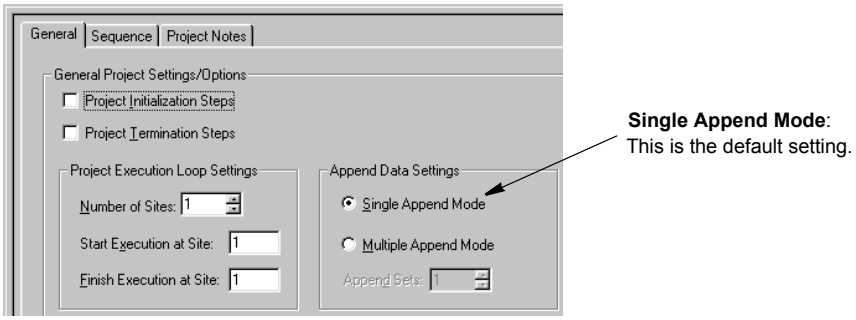
1. In the Project Navigator, open the Project window by double-clicking the Project Plan, as shown in Figure 6-217.

Figure 6-217
Setting the maximum number of Append worksheets



2. In the Project window, under Append Data Settings, do either of the following:
   - If you want to allow only one **Append** worksheet to be generated (for example, if you want to regenerate a single **Append** worksheet multiple times), click the **Single Append Mode** radio button. See Figure 6-218.

Figure 6-218
Allowing only one Append worksheet to be generated or regenerated

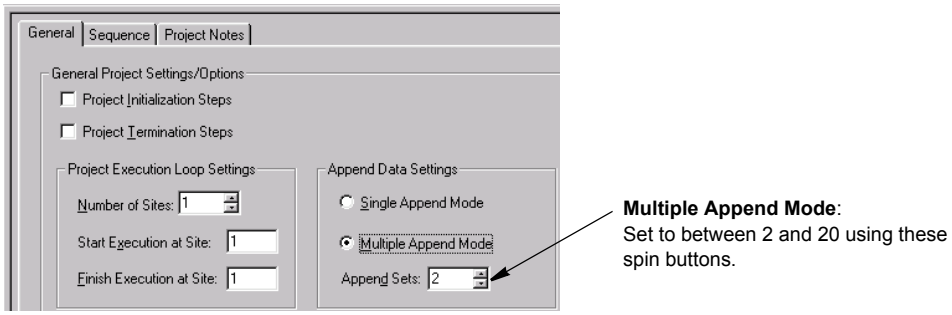


- If you want to allow up to 20 **Append** worksheets to be generated, do the following:

a. Click the **Multiple Append Mode** radio button.

b. Set **Append Sets** to a number between 2 and 20, using the adjacent spin buttons. See Figure 6-219.

Figure 6-219
Allowing multiple Append worksheets to be generated or regenerated



3. At the lower right corner of the Project window, click **Apply**. The new settings take effect.

Performing an Append execution

Generate **Append** data for a test, test sequence, or Project Plan as follows:

1. Initially **Run** execute the test, test sequence, or Project Plan that you want to append, to generate or update the **Data** worksheets for each test (see ***NOTE*** below). For tests and test sequences, refer to the procedure Run execution of individual tests and test sequences. For Project Plans, refer to 'Run' execution of Project Plans.

NOTE *To add **Append** data to an existing test, test sequence, or Project Plan data, do not perform a new **Run** execution. Instead, do the following:*

- *Select a test or test sequence according to "Run execution of individual tests and test sequences," or select a Project Plan and site(s) according to"'Run' execution of Project Plans."*
- *Skip directly to step 2.*

---

2. With step 1 selections still in effect, **Append** execute the test, test sequence, or Project Plan as follows:
   - Click the *green-in-yellow* **Append Data** toolbar button ( ).
   - Select **Append** in the **Run** menu.
   - Simultaneously press the **SHIFT** + **F6** keyboard keys.

CAUTION **Ensure that you use the Append Data function. If you inadvertently use the Run function (for example, by clicking the Run button instead of the Append Data button), you will inadvertently update the Data worksheet(s) and delete all of the Append worksheets for the selected test, test sequence, or Project Plan.**

NOTE *You can subsequently delete **Append** data by a variety of methods, as described in "Deleting Append worksheets." To avoid unwanted data loss, be sure to read and understand the various deletion options before attempting to delete **Append** data.*

Repeating a test

An individual test (ITM or UTM) can be run continuously by using the **Repeat Test Execution** button to start the test. The test does not stop after it is run the first time. Execution continuously loops back to the beginning to keep repeating the test.

Use the **Abort Test/Plan** button to stop the test. When the button is clicked, the test will stop immediately.

The buttons to start repeat test execution and abort the test are shown as follows. Also shown is the button to run the test one time. After using **Repeat**, you can run the test one more time to acquire a complete set of data for the test.

Figure 6-220
Repeating a test

Repeat Test Execution
Click to start continuous run of selected test.

Abort Test/Plan
Click to stop test.

Run Test/Plan
Click to run selected test one time.

Test data

Additional sets of data are NOT generated for repeated tests. When the test repeats, the spreadsheet data for the last test is cleared. A graph will continuously update to reflect the data in the spreadsheet.

When the repeated test is aborted, it will stop immediately. Data for the spreadsheet and graph will be collected up to the point where the test was aborted. If you need a complete set of data for the test, run the test one more time by pressing the green **Run** button, as shown above.

Stress testing

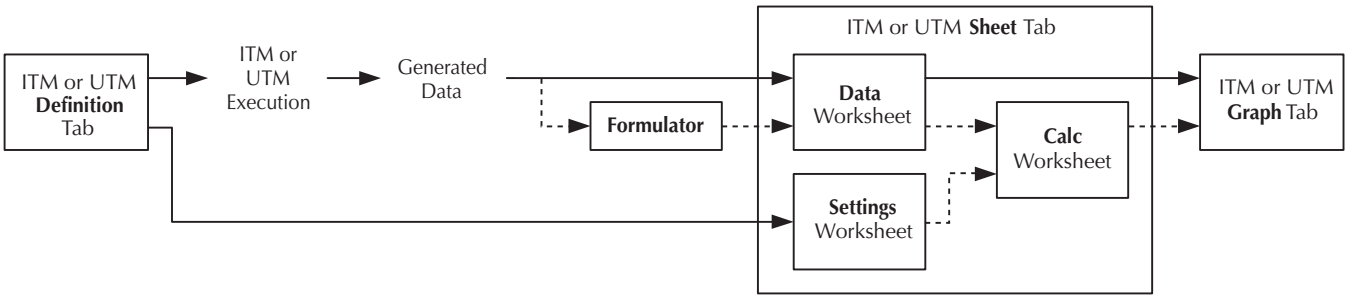
Typical test sequence to use **Repeat** to stress test a device:

1. Run a single test (green **Run** button) to acquire a pre-stress set of data for the device. Make sure to save the data by using the **Save As** button on the **Sheet** tab of the ITM.
2. Use **Repeat** to stress the device over a period of time.
3. When finished with the stressing, use the green **Run** button to run the test one more time to acquire a post-stress set of data.
4. Analyze the affect of stressing by comparing the post-stress data to the pre-stress data.

Displaying and analyzing test results

After you execute Project Plans or individual Subsite Plans, Device Plans, and tests, you can display and analyze test results and test definitions using the following tools: the **Formulator**, the **Sheet** tab worksheets, and the **Graph** tab. Figure 6-221 and the subsequent bulleted list summarize the use of the KITE data display and analysis tools.

Figure 6-221
KITE data display and analysis tools



- Data flow starts at the **Definition** tab of the ITM or UTM. Here, and in accompanying interfaces for an ITM, you configure the test.
- KITE automatically transfers the test configuration to the **Settings** worksheet of the **Sheet** tab for later use. The **Settings** worksheet records all test configuration information in Microsoft Excel compatible format.
- The ITM or UTM executes and generates data, for a device at a specific site.
- KITE inserts the data in the **Data** worksheet of the **Sheet** tab, again in Microsoft Excel-compatible format.
- Optionally KITE extracts additional parameter information from the data through the **Formulator**, using formulas that you previously create. The **Formulator** provides more powerful calculation tools than a spreadsheet (which is also provided in the **Calc** worksheet). KITE performs **Formulator** calculations either immediately after execution or, for some **Formulator** functions used with ITM data, in real time (during test execution).
- KITE inserts the **Formulator** calculation results into the **Data** worksheet, in addition to the raw data.
- Optionally, you can link or paste data or **Formulator** calculations from the **Data** worksheet into the **Calc** worksheet of the **Sheet** tab. Also, you can link entries from the **Settings** worksheet into the **Calc** worksheet. The **Calc** worksheet provides many of the capabilities of a Microsoft Excel or other popular spreadsheet, and its format is Excel compatible. For all cells that 1) contain data or data-derived values from an ITM; and 2) are for calculations, KITE performs real-time calculations, as the test is run.

NOTE *Real-time calculations do not apply to UTM data. A UTM **Data** worksheet does not update until the test is finished.*

- Data and results from both the **Data** worksheet and the **Calc** worksheet can be plotted in the **Graph** tab in a user-specified format, in real time for ITM data and for some ITM **Formulator** calculation results. UTMs can also be plotted in real time (see Enabling real time plotting for UTMs).

Individual subsections below discuss the KITE data viewing and analysis tools in detail.

Displaying and analyzing data using the Sheet tab

The **Sheet** tab of an ITM or UTM window is used to record and manipulate numerical test data and settings. There is a **Sheet** tab corresponding to every ITM/UTM *for each site.* All data in the worksheets of the **Sheet** tab is exportable in Microsoft Excel format.

A **Sheet** tab is effectively a Microsoft Excel compatible workbook that always contains at least the following three worksheets, each of which is subsequently described in more detail:

- **Data** worksheet: The **Data** worksheet of the **Sheet** tab records all of the numerical test data that is generated every time you execute an ITM or a UTM at a given site. The **Sheet** tab **Data** worksheet also records data generated by the **Formulator**.
- **Calc** worksheet: The **Sheet** tab **Calc** worksheet provides a spreadsheet for local data analysis. If there are multiple same-named instances of an ITM or UTM in a Project Plan, the **Calc** worksheet equations are unique for each instance.
- **Settings** worksheet: The **Sheet** tab **Settings** worksheet documents the test configuration and site number.

A **Sheet** tab may also contain one or more **Append** worksheets (**Append1**, **Append2**, ... etc)., as discussed under Append execution of tests, test sequences, and Project Plans. Each **Append** worksheet behaves like a **Data** worksheet. However, its data cannot be plotted on a separate **Graph** tab graph, only on the same graph as the **Data** worksheet data. Refer to Understanding and using the Data worksheet of a Sheet tab.

Each worksheet contains the following controls:

- A data-source identifier.
- The **Save As** button.

Opening a Sheet tab

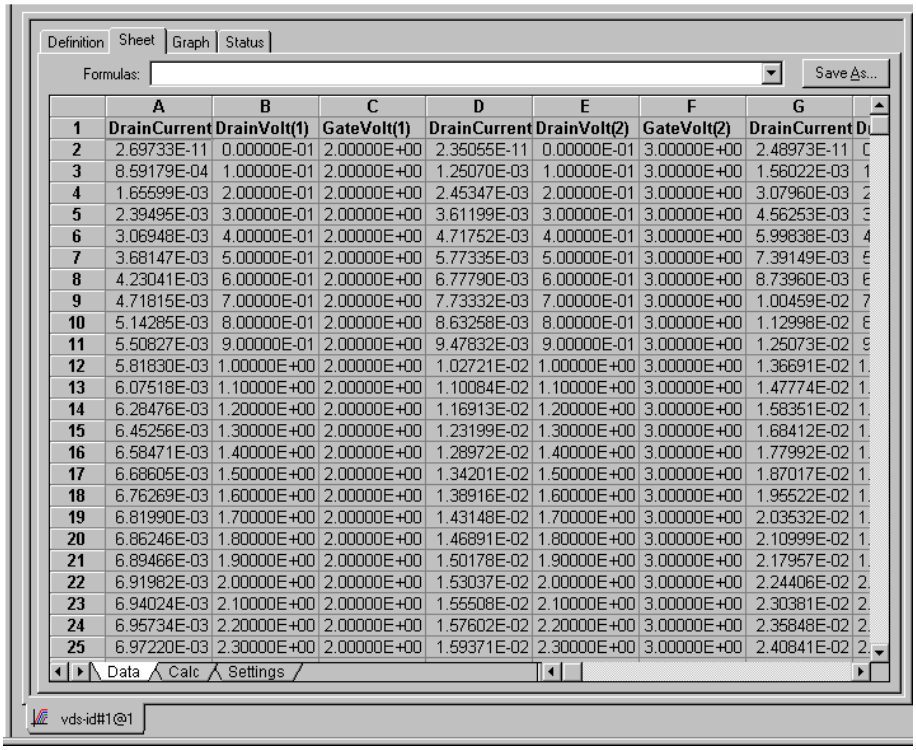
Open a **Sheet** tab as follows:

1. In the Site Navigator, enter the site number where the ITM or UTM was executed, using the spin button controls (the little arrows at the right).
2. In the Project Navigator, double-click the name of the ITM or UTM that acquired the data. An ITM or UTM window appears displaying the **Definition** tab for the selected ITM or UTM.

NOTE *If the Project Plan contains multiple instances of an ITM or UTM under the same name, each instance generates its own data and has its own **UID** (unit identification) number. Ensure that you select the correct instance of the ITM or UTM.*

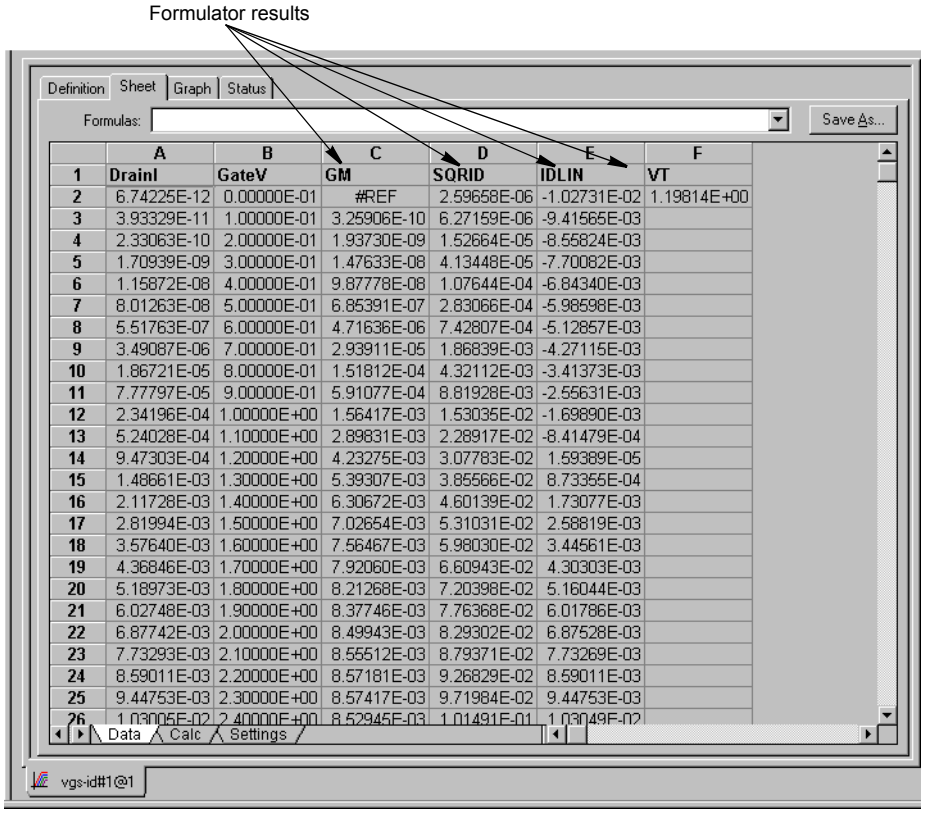
3. Click the ITM or UTM **Sheet** tab. The **Data** worksheet of the **Sheet** tab appears, as well as tabs that provide access to the corresponding **Calc** and **Settings** worksheets.
   Figure 6-222 is the **Data** worksheet of a **Sheet** tab for the **"vds_id"** ITM, showing data for multiple sweeps. Figure 6-223 is the **Data** worksheet of a **Sheet** tab for the **"vgs_id"** ITM, showing **Formulator** calculation results, in addition to test data.

Figure 6-222
Data worksheet of a Sheet tab containing data for multiple sweeps



	A	B	C	D	E	F	G
1	DrainCurrent	DrainVolt(1)	GateVolt(1)	DrainCurrent	DrainVolt(2)	GateVolt(2)	DrainCurrent
2	2.69733E-11	0.00000E-01	2.00000E+00	2.35055E-11	0.00000E-01	3.00000E+00	2.48973E-11
3	8.59179E-04	1.00000E-01	2.00000E+00	1.25070E-03	1.00000E-01	3.00000E+00	1.56022E-03
4	1.65599E-03	2.00000E-01	2.00000E+00	2.45347E-03	2.00000E-01	3.00000E+00	3.07960E-03
5	2.39495E-03	3.00000E-01	2.00000E+00	3.61199E-03	3.00000E-01	3.00000E+00	4.56253E-03
6	3.06948E-03	4.00000E-01	2.00000E+00	4.71752E-03	4.00000E-01	3.00000E+00	5.99838E-03
7	3.68147E-03	5.00000E-01	2.00000E+00	5.77335E-03	5.00000E-01	3.00000E+00	7.39149E-03
8	4.23041E-03	6.00000E-01	2.00000E+00	6.77790E-03	6.00000E-01	3.00000E+00	8.73960E-03
9	4.71815E-03	7.00000E-01	2.00000E+00	7.73332E-03	7.00000E-01	3.00000E+00	1.00459E-02
10	5.14285E-03	8.00000E-01	2.00000E+00	8.63258E-03	8.00000E-01	3.00000E+00	1.12998E-02
11	5.50827E-03	9.00000E-01	2.00000E+00	9.47832E-03	9.00000E-01	3.00000E+00	1.25073E-02
12	5.81830E-03	1.00000E+00	2.00000E+00	1.02721E-02	1.00000E+00	3.00000E+00	1.36691E-02
13	6.07518E-03	1.10000E+00	2.00000E+00	1.10084E-02	1.10000E+00	3.00000E+00	1.47774E-02
14	6.28476E-03	1.20000E+00	2.00000E+00	1.16913E-02	1.20000E+00	3.00000E+00	1.58351E-02
15	6.45256E-03	1.30000E+00	2.00000E+00	1.23199E-02	1.30000E+00	3.00000E+00	1.68412E-02
16	6.58471E-03	1.40000E+00	2.00000E+00	1.28972E-02	1.40000E+00	3.00000E+00	1.77992E-02
17	6.68605E-03	1.50000E+00	2.00000E+00	1.34201E-02	1.50000E+00	3.00000E+00	1.87017E-02
18	6.76269E-03	1.60000E+00	2.00000E+00	1.38916E-02	1.60000E+00	3.00000E+00	1.95522E-02
19	6.81990E-03	1.70000E+00	2.00000E+00	1.43148E-02	1.70000E+00	3.00000E+00	2.03532E-02
20	6.86246E-03	1.80000E+00	2.00000E+00	1.46891E-02	1.80000E+00	3.00000E+00	2.10999E-02
21	6.89466E-03	1.90000E+00	2.00000E+00	1.50178E-02	1.90000E+00	3.00000E+00	2.17957E-02
22	6.91982E-03	2.00000E+00	2.00000E+00	1.53037E-02	2.00000E+00	3.00000E+00	2.24406E-02
23	6.94024E-03	2.10000E+00	2.00000E+00	1.55508E-02	2.10000E+00	3.00000E+00	2.30381E-02
24	6.95734E-03	2.20000E+00	2.00000E+00	1.57602E-02	2.20000E+00	3.00000E+00	2.35848E-02
25	6.97220E-03	2.30000E+00	2.00000E+00	1.59371E-02	2.30000E+00	3.00000E+00	2.40841E-02

Figure 6-223
Data worksheet of a Sheet tab containing both data and Formulator results



	A	B	C	D	E	F
1	DrainI	GateV	GM	SQRID	IDLIN	VT
2	6.74225E-12	0.00000E-01	#REF	2.59658E-06	-1.02731E-02	1.19814E+00
3	3.93329E-11	1.00000E-01	3.25906E-10	6.27159E-06	-9.41565E-03	
4	2.33063E-10	2.00000E-01	1.93730E-09	1.52664E-05	-8.55824E-03	
5	1.70939E-09	3.00000E-01	1.47633E-08	4.13448E-05	-7.70082E-03	
6	1.15872E-08	4.00000E-01	9.87778E-08	1.07644E-04	-6.84340E-03	
7	8.01263E-08	5.00000E-01	6.85391E-07	2.83066E-04	-5.98598E-03	
8	5.51763E-07	6.00000E-01	4.71636E-06	7.42807E-04	-5.12857E-03	
9	3.49087E-06	7.00000E-01	2.93911E-05	1.86839E-03	-4.27115E-03	
10	1.86721E-05	8.00000E-01	1.51812E-04	4.32112E-03	-3.41373E-03	
11	7.77797E-05	9.00000E-01	5.91077E-04	8.81928E-03	-2.55631E-03	
12	2.34196E-04	1.00000E+00	1.56417E-03	1.53035E-02	-1.69890E-03	
13	5.24028E-04	1.10000E+00	2.89831E-03	2.28917E-02	-8.41479E-04	
14	9.47303E-04	1.20000E+00	4.23275E-03	3.07783E-02	1.59389E-05	
15	1.48661E-03	1.30000E+00	5.39307E-03	3.85566E-02	8.73355E-04	
16	2.11728E-03	1.40000E+00	6.30672E-03	4.60139E-02	1.73077E-03	
17	2.81994E-03	1.50000E+00	7.02654E-03	5.31031E-02	2.58819E-03	
18	3.57640E-03	1.60000E+00	7.56467E-03	5.98030E-02	3.44561E-03	
19	4.36846E-03	1.70000E+00	7.92060E-03	6.60943E-02	4.30303E-03	
20	5.18973E-03	1.80000E+00	8.21268E-03	7.20398E-02	5.16044E-03	
21	6.02748E-03	1.90000E+00	8.37746E-03	7.76368E-02	6.01786E-03	
22	6.87742E-03	2.00000E+00	8.49943E-03	8.29302E-02	6.87528E-03	
23	7.73293E-03	2.10000E+00	8.55512E-03	8.79371E-02	7.73269E-03	
24	8.59011E-03	2.20000E+00	8.57181E-03	9.26829E-02	8.59011E-03	
25	9.44753E-03	2.30000E+00	8.57417E-03	9.71984E-02	9.44753E-03	
26	1.03005E-02	2.40000E+00	8.52945E-03	1.01491E-01	1.03049E-02	

NOTE *The **#REF** notation in a cell indicates that a valid value could not be calculated by the **Formulator**. This occurs when a **Formulator** function needs multiple rows as arguments, when a calculated value is out of range, when a divide by zero is attempted, and so on.*

*In the **GM** column in Figure 6-223, note the **#REF** notation in the first row. Each value in the **GM** column is a difference coefficient that is calculated as the ratio **ΔDrainI** /**ΔGateV**, where **ΔDrainI** and **ΔGateV** are differences between values in the present row and values in the previous row. Because, no previous row exists before the first row, a valid calculation is not possible for the first row. Therefore, the **Formulator** returns the **#REF** notation.*

*A column will contain multiple instances of **#REF** if the **Formulator** function requires multiple prior cells for the calculation. For example, if the MAVG function is using five data points to calculate a moving average of a column containing five values, the first two and last two cells will contain **#REF**.*

Understanding and using the Data worksheet of a Sheet tab

The **Data** worksheet first appears when you open the **Sheet** tab (see Figures 6-222 and 6-223). The **Data** worksheet displays all the data that was last generated by the ITM or UTM for a particular site. The **Data** worksheet also contains the results of any **Formulator** calculations that were performed on the last-generated data. Features of the **Data** worksheet are as follows:

- Data is reported in Microsoft Excel compatible format, each column containing the results for one test parameter or for a **Formulator** calculation.

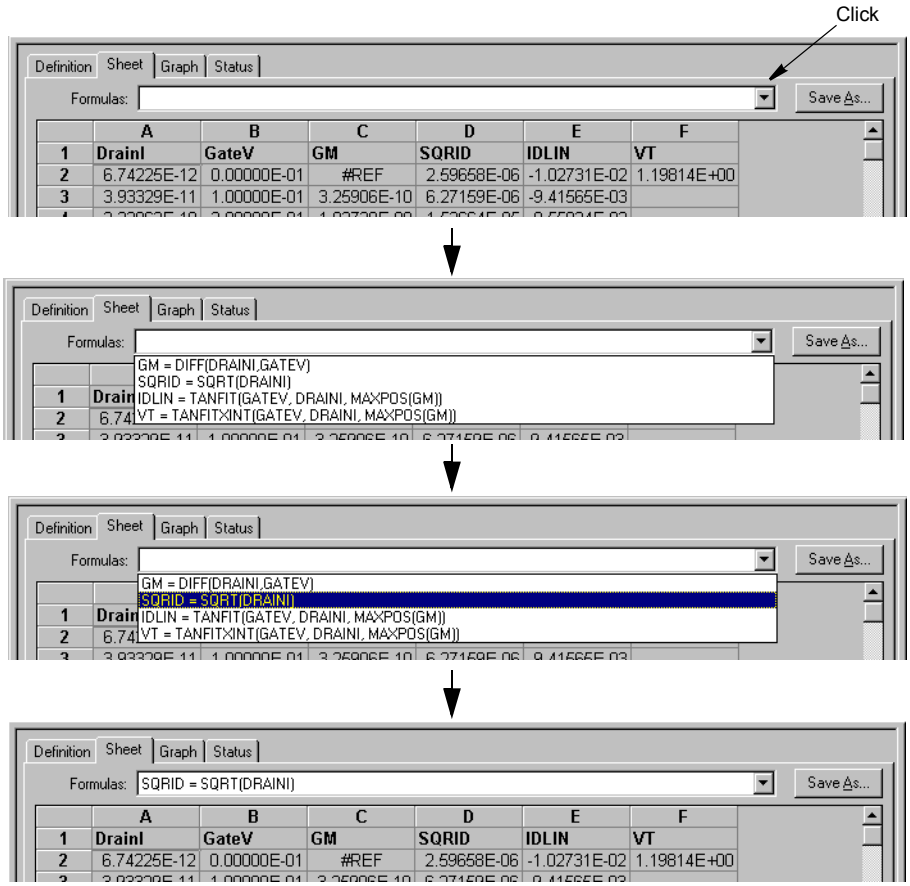
NOTE *Some **Formulator** calculations return only a single value.*

  The display of all columns for the test may span several pages horizontally. The display of data in a given column may span several pages vertically.
- Column headings, each of which identifies the data below it by one of the following:
  - The name of a test-results parameter (e.g., current or voltage) that is assigned by KITE, by the user (for an ITM only), or by the KULT programmer (for a UTM only). For ITM current and voltage naming, refer to Understanding and configuring the Measuring Options area.
  - The name of a **Formulator** results parameter.
- The data-source identifier, the **Formula** combo box, and the **Save As** button, each of which are discussed below.
- The contents of the **Data** worksheet are display-only. However, you can manipulate the contents of the **Data** worksheet after linking it to or pasting it in the **Calc** worksheet.

Understanding the Formula combo box of the Data worksheet

If a column in the **Data** worksheet contains the results of **Formulator** calculations, you can locally display the formula (equation) that was used to obtain the results. Display the formula by selecting it from the **Formula** combo box, as illustrated in Figure 6-224. The steps in Figure 6-224 display the formula that was used to obtain the **SQRID** results in Figure 6-223.

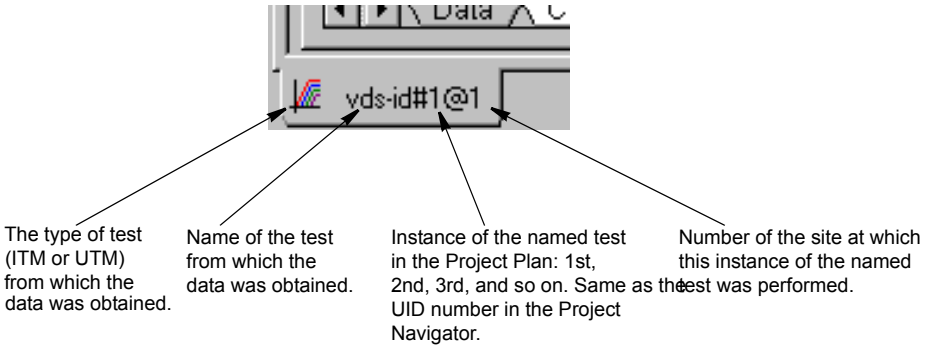
Figure 6-224
Displaying a Formulator equation using the Formula combo box



Understanding the data-source identifier

The ITM or UTM window tab at the bottom of all **Sheet** tab windows identifies the source of the data in the **Sheet** tab, as shown in Figure 6-225.

Figure 6-225
Data-source identifier



Saving a worksheet

Saving a Sheet tab to the Project Plan

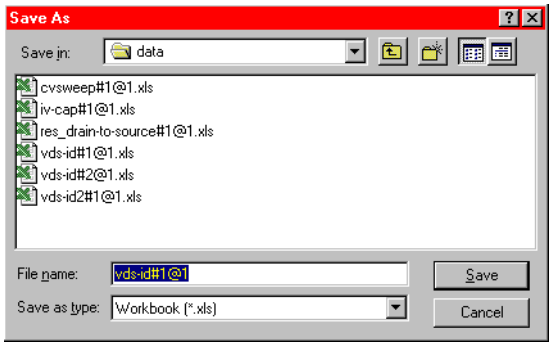
To save the displayed data to the Project Plan, do one of the following: click **Save** in the **File** menu, click the single floppy-disk toolbar button, or enter `[CNTRL + S]` at the keyboard.

Saving the Sheet tab to an external spreadsheet file using the Save As button

All data in the **Sheet** tab for a test is in Microsoft Excel compatible format, with the.xls extension. In other words, the combined worksheets in the Sheet tab (including any Append1, Append 2, and so on worksheets) effectively comprise a workbook that can be used directly in an Excel compatible spreadsheet program. To save the contents of all **Sheet** tab worksheets to a designated folder simultaneously in a single `.xls` file, do the following:

1. Click **Save As** in the upper right corner of any of the three worksheets. The Save As window displays, with **Workbook (*.xls)** as the default file type. See Figure 6-226.

Figure 6-226
Data Save As window, configured for workbook files



2. In the **Save In** edit box of the Save As window, select the location for the text file.
3. In the **File name** edit box of the Save As window, Keithley recommends that you retain the default selection, which contains the data-source identifier (refer to Understanding the data-source identifier).
4. In the **Save as type** combo box, make no changes; retain the `*.xls` type.
5. Click **Save.**

NOTE *Do not attempt to use the **Save As** button to save data to the Project Plan.*

Saving a Sheet tab worksheet to a tab delimited text file using the Save As button

Worksheets can also be saved individually, only, as formatted, **tab delimited** text files. Figure 6-227 illustrates the **Data** worksheet of Figure 6-222 saved as a text file and displayed in Windows Notepad.

Figure 6-227
Example of a Data worksheet saved as a tab delimited text file

vds-id#1@1-Dat.txt - Notepad

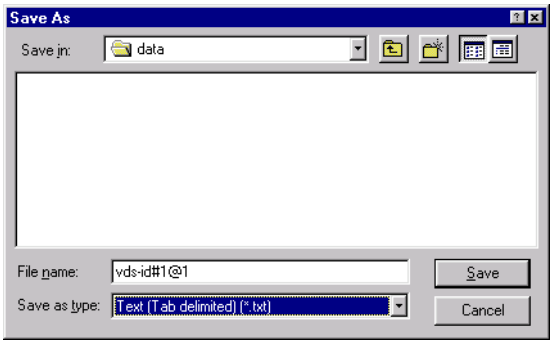
File Edit Search Help

DrainCurrent(1)	DrainVolt(1)	GateVolt(1)	DrainCurrent(2)	DrainVolt(2)	GateVolt(2)
2.69733E-11	0.00000E-01	2.00000E+00	2.35055E-11	0.00000E-01	3.00000E+00
8.59179E-04	1.00000E-01	2.00000E+00	1.25070E-03	1.00000E-01	3.00000E+00
1.65599E-03	2.00000E-01	2.00000E+00	2.45347E-03	2.00000E-01	3.00000E+00
2.39495E-03	3.00000E-01	2.00000E+00	3.61199E-03	3.00000E-01	3.00000E+00
3.06948E-03	4.00000E-01	2.00000E+00	4.71752E-03	4.00000E-01	3.00000E+00
3.68147E-03	5.00000E-01	2.00000E+00	5.77335E-03	5.00000E-01	3.00000E+00
4.23041E-03	6.00000E-01	2.00000E+00	6.77790E-03	6.00000E-01	3.00000E+00
4.71815E-03	7.00000E-01	2.00000E+00	7.73332E-03	7.00000E-01	3.00000E+00
5.14285E-03	8.00000E-01	2.00000E+00	8.63258E-03	8.00000E-01	3.00000E+00
5.50827E-03	9.00000E-01	2.00000E+00	9.47832E-03	9.00000E-01	3.00000E+00
5.81830E-03	1.00000E+00	2.00000E+00	1.02721E-02	1.00000E+00	3.00000E+00
6.07518E-03	1.10000E+00	2.00000E+00	1.10084E-02	1.10000E+00	3.00000E+00
6.28476E-03	1.20000E+00	2.00000E+00	1.16913E-02	1.20000E+00	3.00000E+00
6.45256E-03	1.30000E+00	2.00000E+00	1.23199E-02	1.30000E+00	3.00000E+00
6.58471E-03	1.40000E+00	2.00000E+00	1.28972E-02	1.40000E+00	3.00000E+00
6.68605E-03	1.50000E+00	2.00000E+00	1.34201E-02	1.50000E+00	3.00000E+00
6.76269E-03	1.60000E+00	2.00000E+00	1.38916E-02	1.60000E+00	3.00000E+00
6.81990E-03	1.70000E+00	2.00000E+00	1.43148E-02	1.70000E+00	3.00000E+00

To save the contents of the currently displayed worksheet to a designated folder as a tab delimited text (.txt) file, do the following:

1. Click **Save As** in the upper right corner of the worksheet. The Save As window displays.
2. In the **Save as type** combo box, select **Text (Tab delimited) (*.txt)**. See Figure 6-228.

Figure 6-228
Data Save As window configured for tab delimited text files



3. In the **Save in** edit box of the Save As window, select the location for the tab delimited text file.
4. In the **File name** edit box of the **Save As** window, Keithley recommends that you *add* a modifier to the displayed file name. In that way, the name both retains the data-source identifier and identifies the worksheet type. For example, you might save the vds-id#1@1 worksheets as follows:
   - The **Data** worksheet as **vds-id#1@1-Dat**.
   - The **vds-id#1@1 Calc** worksheet as **vds-id#1@1-Clc**.
   - The **vds-id#1@1 Settings** worksheet as **vds-id#1@1-Stg**.
   - The **Append** worksheets, if any, as **vds-id#1@1-Ap1**, **vds-id#1@1-Ap2**,... and so on.

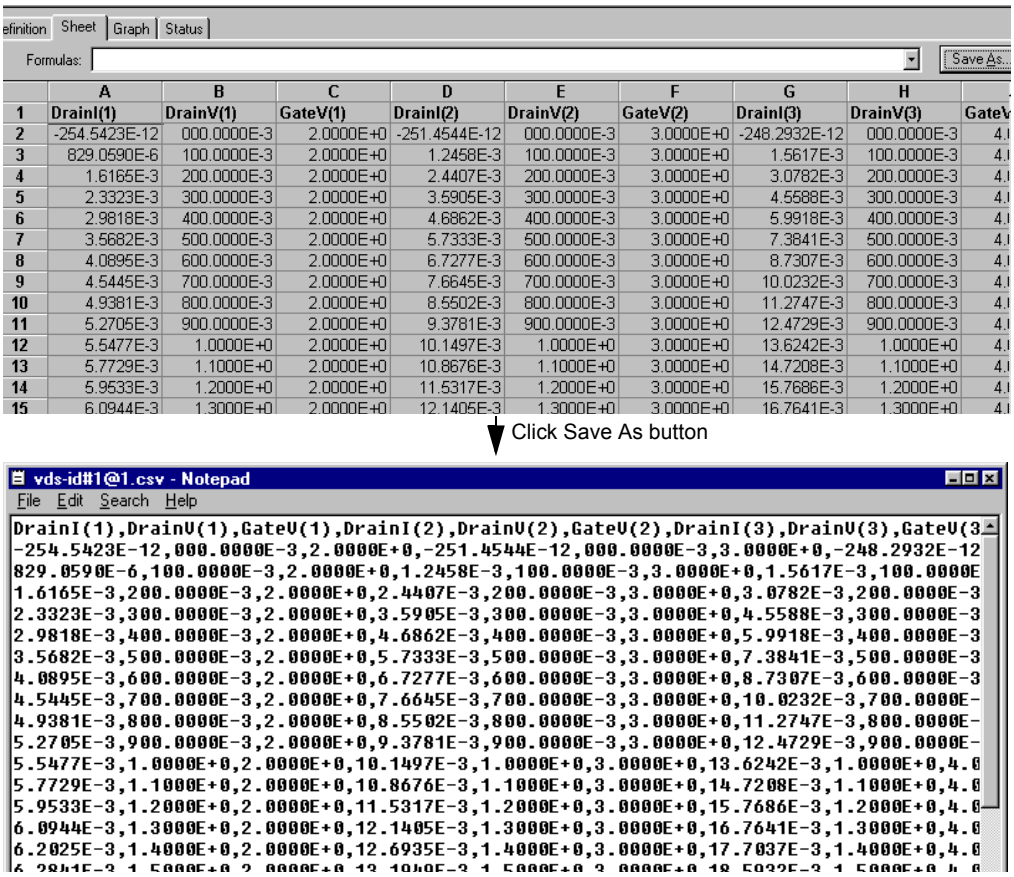
CAUTION **For a given test, the *default* filename that is displayed in the "File name" box (the data-source-identifier name) is the same for all of the worksheets. However, when you specify the tab delimited text format (*.txt), KITE attempts to save each of the worksheets as a separate file. Therefore, you must give each of the worksheets a separate filename (preferably by adding a worksheet-specific modifier to the default filename). Otherwise, the save of one worksheet will overwrite the save of another.**

5. Click **Save.**

Saving a Sheet tab worksheet to a comma delimited text file using the Save As button

Worksheets can also be saved individually, o*nly,* as comma delimited text files. Figure 6-229 illustrates a **vds-id Data** worksheet saved as a comma delimited text file and displayed in Windows Notepad.

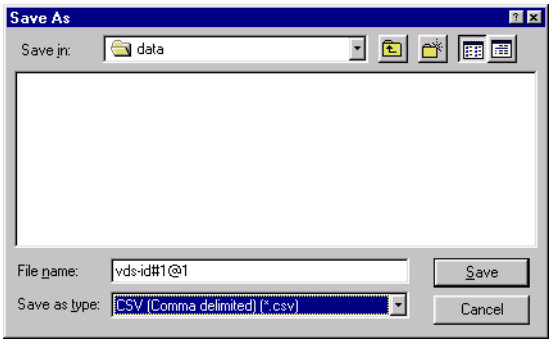
Figure 6-229
Example of a Data worksheet saved as a comma-delimited text file



To save the contents of the currently displayed worksheet to a designated folder as a comma delimited text file (.csv file extension), do the following:

1. Click **Save As** in the upper right corner of the worksheet. The Save As window displays.
2. In the **Save as type** combo box, select **CSV (Comma delimited) (*.csv)**. See Figure 6-230.

Figure 6-230
Data Save As window configured for comma delimited text files



3. In the **Save in** edit box of the Save As window, select the location for the comma delimited text file.
4. In the **File name** edit box of the **Save As** window, Keithley recommends that you add a modifier to the displayed file name. In that way, the name both retains the data-source identifier and identifies the worksheet type. For example, you might save the vds-id#1@1 worksheets as follows:
   - The **Data** worksheet as **vds-id#1@1-Dat**.
   - The **vds-id#1@1 Calc** worksheet as **vds-id#1@1-Clc**.
   - The **vds-id#1@1 Settings** worksheet as **vds-id#1@1-Stg**.
   - The **Append** worksheets, if any, as **vds-id#1@1-Ap1**, **vds-id#1@1-Ap2**,... and so on.

CAUTION **For a given test, the default filename that is displayed in the "File name" box (the data-source-identifier name) is the same for all of the worksheets. However, when you specify the comma-separated-values format (*.csv), KITE attempts to save each of the worksheets as a separate file. Therefore, you must give each of the worksheets a separate filename (preferably by adding a worksheet-specific modifier to the default filename). Otherwise, the save of one worksheet will overwrite the save of another.**

5. Click **Save.**

Understanding and using Append worksheets of a Sheet tab

The optional **Append** feature appends (layers) curves from multiple runs on a single graph and creates a separate worksheet for each **Append** execution. Refer also to Append execution of tests, test sequences, and Project Plans and Appending curves from multiple runs on a single graph

Understanding Append worksheets

The following applies to the worksheets that are created by **Append** executions:

- The data generated for each **Append** execution of a test is located in an individual **Append** worksheet, where **n** designates the nth **Append** execution. That is, the worksheets are labeled **Append1**, **Append2**, ... and so on.

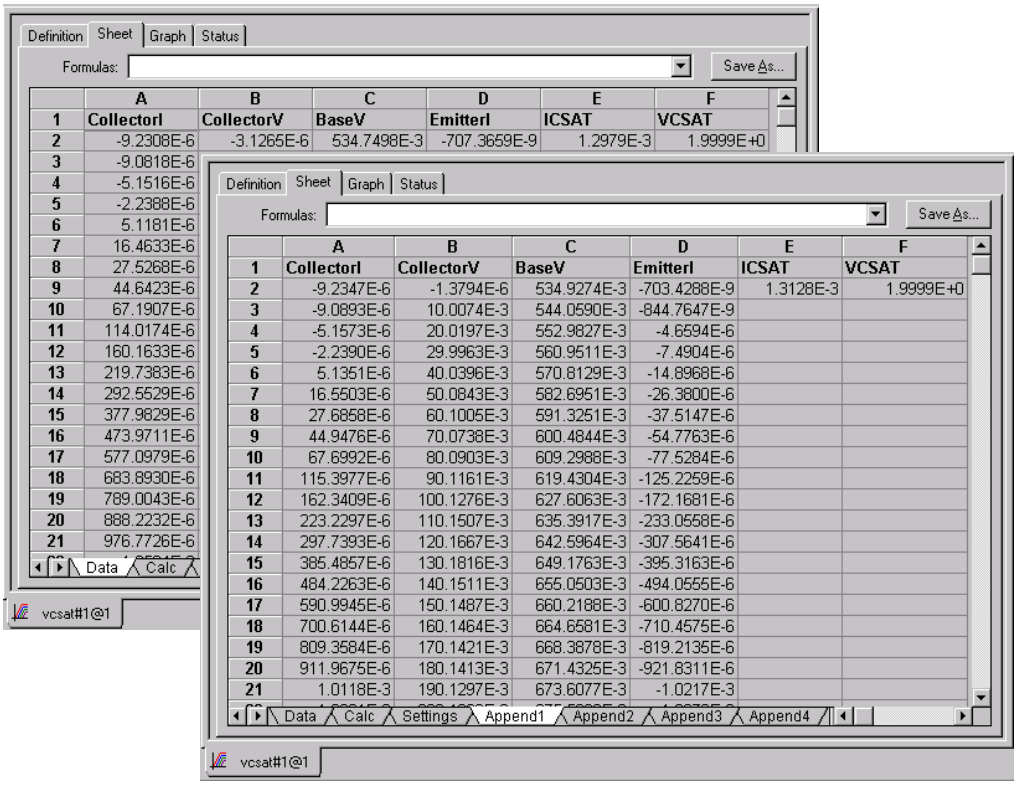
NOTE *You can specify the maximum number of **Append** executions and worksheets (the maximum value of "**n**"). After the maximum number of **Append** worksheets have been generated, the data from each **Append** execution replaces the data from the previous **Append** execution. For example, if the maximum value of "**n**" is 4, the data*

*from the fifth **Append** execution replaces the data from the fourth **Append** execution. Refer also to Append execution of tests, test sequences, and Project Plans*

- Each **Append** worksheet is labeled with a separate tab to distinguish it from the **Data** worksheet for the test.
- Each **Append** worksheet contains the same columns and rows as the **Data** worksheet for the test.
- Each **Append** worksheet may be manipulated in the same way as the **Data** worksheet for the test.

See Figure 6-231.

Figure 6-231
Data and Append1 worksheets for a particular vcsat test



NOTE *To display hidden **Append** worksheet tabs, use the scroll buttons located at the left side of the tabs:*

Figure 6-232
Append worksheet tabs

Append executions are not restricted to individual tests. An entire test sequence (Device Plan or Subsite Plan) or a Project Plan may be **Append** executed **n** times, resulting in **n** separate **Append** worksheets for each test in the sequence or Project Plan. Multi-site **Append** execution of a Project Plan results in multi-level sets of **Append** worksheets.

Graphing the Append worksheet data

You can graph **Append** worksheet data in essentially the same way as **Data** worksheet data. Refer to Appending curves from multiple runs on a single graph.

Deleting Append worksheets

You can delete **Append** worksheets using the following three methods:

- **Clear Append Data method:** Involves the **Clear Append Data** toolbar button/menu item.
- **Run method**: Involves performing a **Run** execution.
- **Append Sets method**: Involves reducing the Project window **Append Sets** value.

The next three subsections outline advantages, disadvantages, and procedures for each method.

NOTE *It is not possible to delete individual **Append** worksheets for a specific test, a test sequence, or a Project Plan (though it is possible to simultaneously delete a group of the highest numbered Append worksheets for all tests in a Project Plan at all sites). However, you can choose to exclude specific **Append** worksheet data from a graph. Refer to Append selections in the Graph Definition window.*

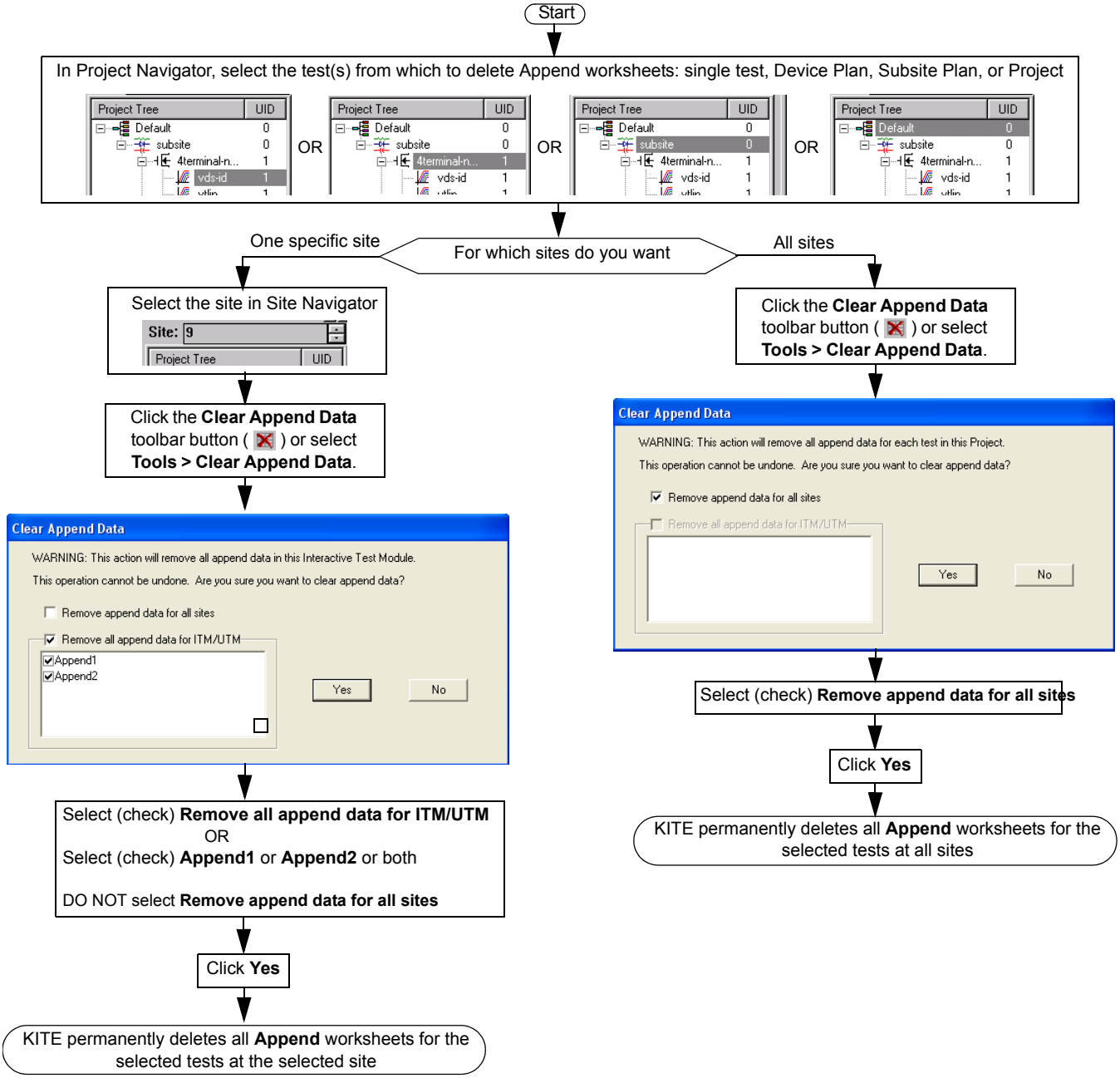
Clear Append Data method for deleting Append worksheets

Use the **Clear Append Data** function to *permanently* delete any or all **Append** worksheets for a selected test, test sequence, or Project Plan, either at one specific site or at all sites.

- **Advantages**:
  - Perhaps the easiest, most straightforward method.
  - Deletes **Append** worksheets without modifying the **Data** worksheet(s).
- **Disadvantages**:
  - Final. Recovery from accidental deletion is not possible.

The Clear Append Data method is explained in Figure 6-233. If there is no append data for an ITM or UTM, the append list will be blank and the selection boxes for ITM/UTM append data will be disabled.

Figure 6-233
Clear Append Data-method procedure

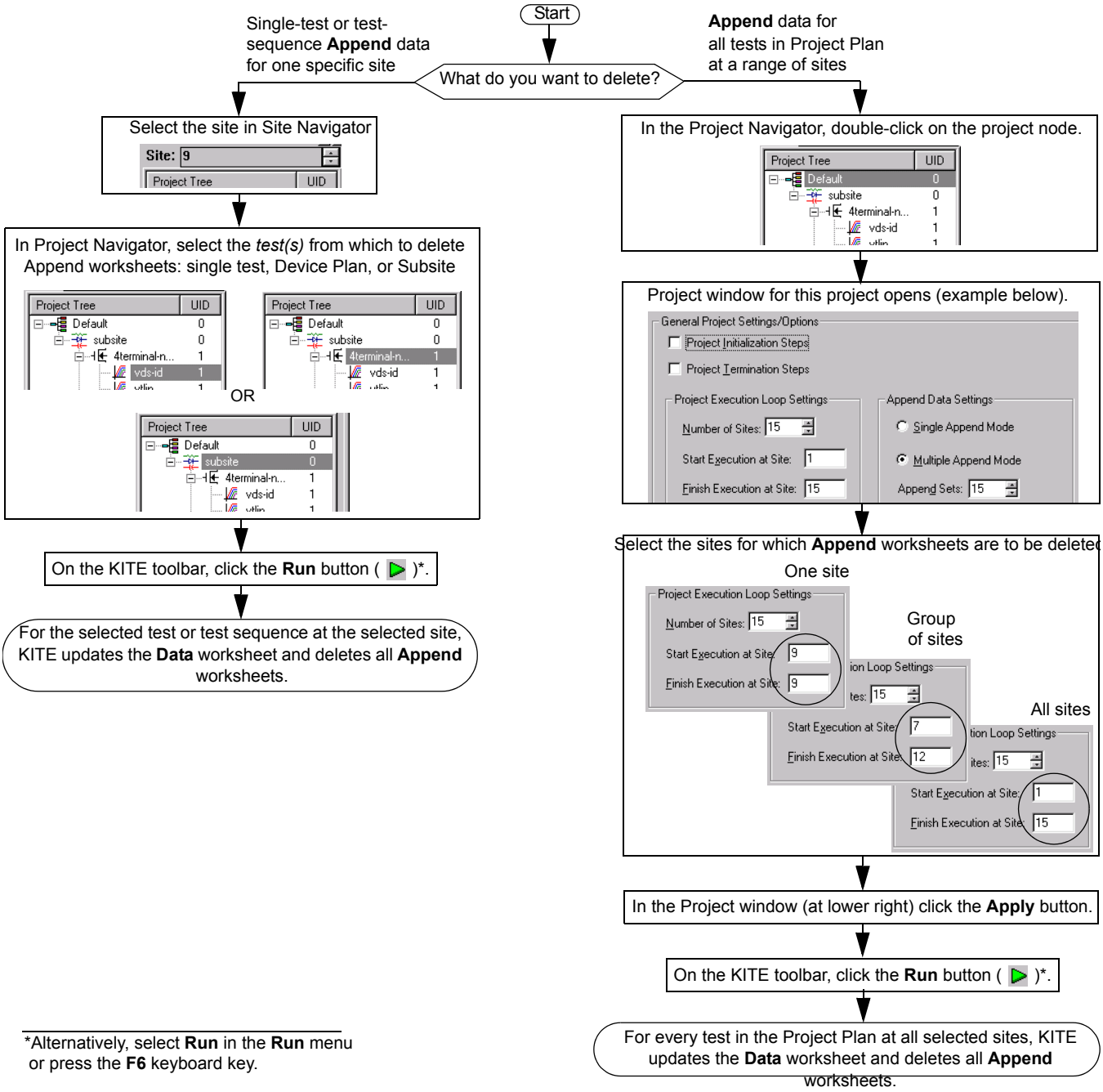


***Run** method for deleting **Append** worksheets*

For selected test(s) and sites, perform a **Run** execution to update the corresponding **Data** worksheet(s) and simultaneously delete all **Append** worksheets.

- **Advantage**:
  - Deletes **Append** worksheets for selected range of sites (for full Project Plan).
- **Disadvantages**:
  - Cannot delete **Append** worksheets without updating the corresponding **Data** worksheets (however, this characteristic can be an advantage in some situations).
  - Can only delete *all* **Append** worksheets for a test.

Figure 6-234
Run-method procedure



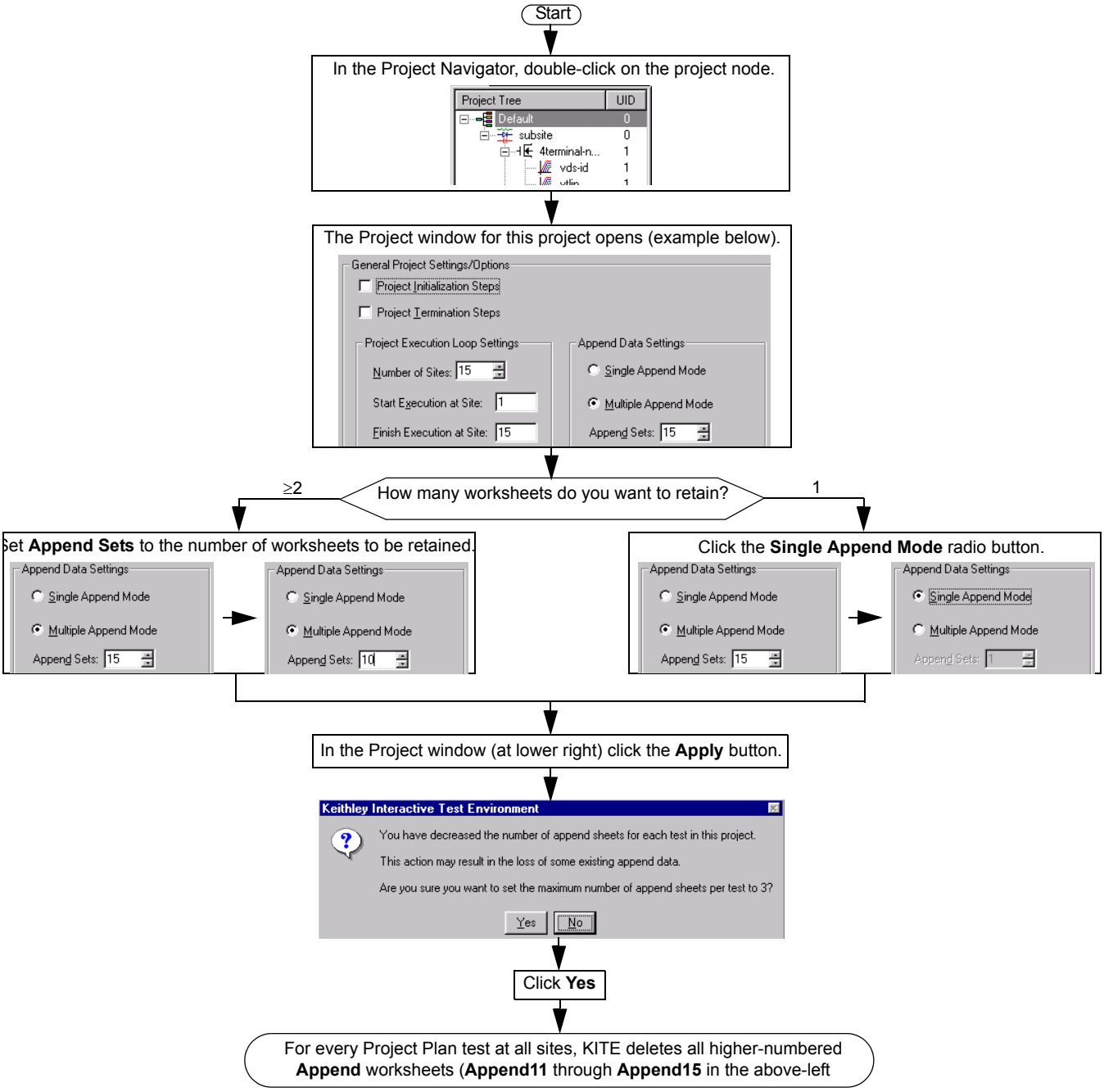
Append Sets method for deleting Append worksheets

Reduce the **Append Sets** value in the Project window to reduce the number of **Append** worksheets per test, for all tests in the Project Plan at all sites. For example, suppose the **Append Sets** value (the maximum number of **Append** worksheets) is initially set to 15. If you reduce **Append Sets** to 10 and then click **Apply**, KITE deletes any existing **Append11, Append12, Append13**, **Append14**, and **Append15** worksheets.

- **Advantage**: Can delete a selected upper range of **Append** worksheets.

- **Disadvantage**: Cannot delete **Append** worksheets for selected tests or sites. You can delete the **Append** worksheets only for all Project Plan tests at all sites.

Figure 6-235
Append Sets-method procedure



Understanding and using the Calc worksheet of a Sheet tab

The **Calc** worksheet is a Microsoft Excel compatible spreadsheet that provides many of the capabilities of popular spreadsheets.

NOTE *Use of the **Calc** sheet requires a working knowledge of Excel or a similar spreadsheet.*

The **Calc** worksheet allows you to:

- Hot-link and copy values and information from the **Data** and **Settings** worksheets.
- Perform additional data analysis or scratch pad calculations.
- Graph the calculation results using the **Graph** tab (any **Calc** worksheet column with an entry in the first row is automatically available in the **Graph** tab as a potential plot variable. Refer to Displaying and analyzing data using the Sheet tab).

For all cells that 1) contain hot-linked data or data-derived values from an ITM; and 2) are for calculations, KITE performs real-time calculations, as the test is run.

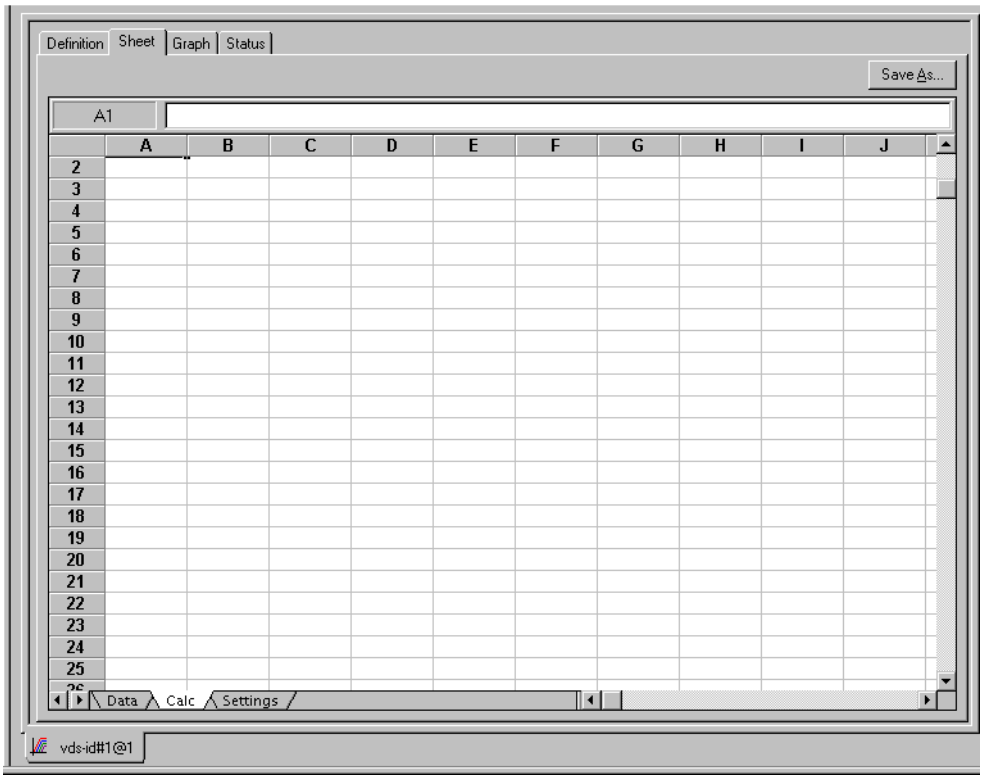
NOTE *Real-time calculations do not apply to UTM data. A UTM **Data** worksheet does not update until the test is finished.*

Opening a Calc worksheet

Open a **Calc** worksheet as follows:

- Open the **Sheet** tab for the test, as described in Opening a Sheet tab.
- Click the **Calc** label at the bottom of the **Sheet** tab. The **Calc** worksheet appears, as shown in Figure 6-236.

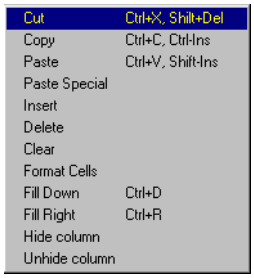
Figure 6-236
Calc worksheet



Opening the Calc sheet pop-up menu

A pop-up menu appears when you right-click in the Calc sheet. See Figure 6-237.

Figure 6-237
Calc worksheet pop-up menu



Hot-linking Data and Settings worksheet cells to Calc worksheet cells

The **Calc** worksheet follows all the rules of Excel with regard to cell references between worksheets. Therefore, you can hot-link a **Calc** worksheet cell to any **Data** worksheet or **Settings** worksheet cell. When the contents of hot-linked **Data** or **Settings** worksheet cells change, the content of the corresponding **Calc** worksheet cells change identically.

Hot-link the contents of a worksheet cell(s) as follows:

1. Click in the **Calc** worksheet cell where you want to link to a **Data** or **Settings** worksheet value.
2. Do one of the following:
   - To link to a cell in the **Data** sheet, type in **=data!CellNumber**.
   - To link to a cell in the **Settings** worksheet, type in **=settings!CellNumber**.
     where **CellNumber** is the number of the single **Data** or **Settings** cell that you want to hot-link to.
3. Press the **ENTER** key. The formula is replaced by the hot-linked data from the **Data** or **Settings** worksheet.

Thereafter, by dragging, you can hot-link to the contents of a range of the **Data** or **Settings** cells immediately below or to the right of the **Data** or **Setting** worksheet cell that you just hot-linked to. To accomplish this, do the following:

1. Select the **Calc** worksheet cell containing the hot-linked data, as inserted using steps 1 through 3 above.
2. Move the cursor to the lower right corner of this cell until you see the small black plus sign.
3. When the small black plus sign appears at the corner, do one of the following:
   - To link a below-adjacent (higher-numbered) range of **Data** or **Settings** column cells, press the left mouse button and drag the small black plus sign down.
   - To link a right-adjacent range of **Data** or **Setting**s row cells, press the left mouse button and drag the small black plus sign to the right.

Setting up calculations in a Calc worksheet

You enter formulas and perform calculations in the **Calc** worksheet essentially the same as you would in an Excel or other popular spreadsheet. The **Calc** worksheet is provided under the assumption that most users are already familiar with the use of spreadsheets. However, if you are unfamiliar with spreadsheets, Keithley Instruments suggests that you review one of the many excellent manuals available on the subject.

In any case, before performing calculations with the **Calc** worksheet, review the available **Calc** mathematical functions in the next subsection.

Understanding the supported Calc worksheet functions

Keithley supports a variety of **Calc** worksheet functions, which are used in the same way as typical spreadsheet functions. These functions are identified below, in terms of purpose, format, and required arguments. An example is given in each case.

NOTE *In the **Calc** worksheet functions, 10 point Courier distinguishes function format parameter names from other text.*

ABS: Calc worksheet function

Purpose Returns the absolute value of a value

Format `ABS (Value)`

Where: `Value =` Any number

Example Both `ABS(1)` and `ABS(-1)` return a value of 1.

Remarks An absolute value does not display a positive or negative sign.

See also **`SIGN Calc worksheet function.`**

ACOS: Calc worksheet function

Purpose Returns the arc cosine of a value.

Format `ACOS (Value)`

Where: `Value =` The cosine of an angle, within the range +1 to -1

Example `ACOS(0.5)` returns 1.05.
`ACOS(-0.2)` returns 1.77.

Remarks The resulting angle is returned, in radians (from 0 to π). To convert the result in radians to a result in degrees, multiply the result in radians by `180/PI()`.

See also **`COS Calc worksheet function.`**

ACOSH: Calc worksheet function

Purpose Returns the inverse hyperbolic cosine of a value.

Format `ACOSH(Value)`

Where: `Value =` Any number equal to or greater than 1.

Example `ACOSH(1.2)` returns 0.62.

`ACOSH(3)` returns 1.76.

See also **ASINH, ATANH, and COSH Calc worksheet functions.**

ASIN: Calc worksheet function

Purpose Returns the arcsine of a value.

Format `ASIN(Value)`

Where: *Value* = The sine of the resulting angle, ranging from -1 to 1.

Example `ASIN(1)` returns 1.57.
`ASIN(0.4)` returns 0.41.

Remarks The resulting angle is returned in radians (ranging from $-\pi/2$ to $\pi/2$). To convert the result in radians to a result in degrees, multiply the result in radians by `180/PI()`.

See also **ASINH, PI, and SIN Calc worksheet functions.**

ASINH: Calc worksheet function

Purpose Returns the inverse hyperbolic sine of a value.

Format `ASINH(Value)`

Where: *Value* = Any number.

Example `ASINH(5.3)` returns 2.37.
`ASINH(-4)` returns -2.09.

See also **ACOSH, ASIN, ATANH, and SINH Calc worksheet functions.**

ATAN: Calc worksheet function

Purpose Returns the arctangent of a number.

Format `ATAN(Value)`

Where: *Value* = The tangent of the resulting angle.

Example `ATAN(3.5)` returns 1.29.
`ATAN(4)` returns -1.33.

Remarks The resulting angle is returned in radians (ranging from $-\pi/2$ to $\pi/2$). To convert the result in radians to a result in degrees, multiply the result in radians by `180/PI()`.

See also **ATAN2, ATANH, PI, and TAN Calc worksheet functions.**

ATAN2: Calc worksheet function

Purpose Returns the arctangent of specified coordinates (see Remarks).

Format `ATAN2(x, y)`

Where: *x* = The x coordinate
y = The y coordinate

Example `ATAN2(3, 6)` returns 1.11
`ATAN2(-1, 0.1)` returns 3.04

Remarks The arctangent is the angle between the x axis and a line having the following end points:

The origin (0, 0)

The point at the coordinates (*x*,*y*)

The angle is returned in radians, ranging *between* -π and π (-π is excluded).

See also **ATAN, ATANH, PI, and TAN Calc worksheet functions.**

ATANH: Calc worksheet function

Purpose Returns the inverse hyperbolic tangent of a number.

Format `ATANH(Value)`

Where: *Value* = A number between -1 and 1, excluding -1 and 1.

Example `ATANH(0.5)` returns 0.55.
`ATANH(-0.25)` returns -0.26.

See also **ACOS, ASINH and TANH Calc worksheet functions.**

AVERAGE: Calc worksheet function

Purpose Returns the average of the supplied numbers. The result of AVERAGE is also known as the arithmetic mean.

Format `AVERAGE(Value_list)`

Where: *Value_list* = A list of numbers separated by commas or a range of number-containing cells in the **Calc** worksheet. As many as 30 numbers can be averaged. Text, logical expressions, or empty cells in a cell range are ignored. However, all numeric values are used, including 0.

Example `AVERAGE(5, 6, 8, 14)` returns 8.25.
`AVERAGE(C15:C17)` returns 134, the average of the values in cells C15. through C17 of a particular **Calc** worksheet.

See also **MIN and MAX Calc worksheet functions.**

COS: Calc worksheet function

Purpose Returns the cosine of an angle.

Format `COS(Value)`

Where: *Value* = The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying it by `PI/180`.

Example `COS(1.4444)` returns 0.126.
`COS(5)` returns 0.28.

See also **ACOS, ASINH, ATANH, COSH and PI Calc worksheet functions.**

COSH: Calc worksheet function

Purpose Returns the hyperbolic cosine of an angle.

Format `COSH(Value)`

Where: *Value* = Any value.

Example `COSH(2.10)` returns 4.14.
`COSH(0.24)` returns 1.03.

See also **ASINH, ATANH and COS Calc worksheet functions.**

DAY: Calc worksheet function

Purpose Returns the day-of-the-month component of the supplied date/time serial number.

Format `DAY(Serial_number)`

Where: *Serial_number* = A date represented as a serial number or text (for example, 06-21-94 or 21-Jun-94).

Example `DAY(34399)` returns 6.
`DAY("06-21-94")` returns 21.
`DAY(NOW())` returns the present day of the month.

Remarks Needed to extract the day from the serial number created by the **NOW** function.

See also **HOUR, MINUTE, MONTH, NOW, SECOND, and YEAR Calc worksheet functions.**

EXP: Calc worksheet function

Purpose Returns the constant e raised to the specified power. The constant e is 2.71828182845904 (the base of the natural logarithm).

Format `EXP(Value)`

Where: *Value* = Any number as the exponent.

Example `EXP(2.5)` returns 12.18.
`EXP(3)` returns 20.09.

See also **LN and LOG Calc worksheet functions.**

FIXED: Calc worksheet function

Purpose Rounds a number to the supplied precision, formats the number in decimal format and returns the result as text.

Format `FIXED(Value [, Precision][, No_commas])`

Where: *Value* = Any number.
Precision = The number of digits that appear to the right of the decimal point. When this argument is omitted, a default precision of 2 is used. If you specify negative precision, *Value* is rounded to the left of the decimal point. You can specify a precision as great as 127 digits.
No_commas = Determines if thousands separators (commas) are used in the result. Use 1 to exclude commas in the result. If *No_commas* is 0 or the argument is omitted, thousands separators are included (for example, 1,000.00).

Example `FIXED(2000.5, 3)` returns 2,000.500.
`FIXED(2009.5, -1,1)` returns 2010.

See also **DOLLAR, ROUND, TEXT, and VALUE Calc worksheet functions.**

HOUR: Calc worksheet function

Purpose Returns the hour component of the supplied date/time serial number, specified in 24-hour format.

Format `HOUR(Serial_number)`

Where: *Serial_number* = The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Example `HOUR(34259.4)` returns 9.
`HOUR(34619.976)` returns 23.
`HOUR(NOW())` returns the present hour of the present day.

Remarks The result is an integer ranging from 0 (12:00 AM) to 23 (11:00 PM).

Needed to extract the hour from the serial number created by the **NOW** function.

See also **DAY, MINUTE, MONTH, NOW, SECOND, and YEAR Calc worksheet functions.**

IF: Calc worksheet function

Purpose Tests the condition and returns the specified value.

Format `IF(Condition, True_number, False_number)`

Where: *Condition* = Any logical expression.
True_number = The value to be returned if *Condition* evaluates to True.
False_number = The value to be returned if *Condition* evaluates to False.
`IF(A1>10, "Greater", "Less")` returns `Greater` if the contents of `A1` is greater than 10 and `Less` if the contents of A1 is less than 10.

LN: Calc worksheet function

Purpose Returns the natural logarithm (based on the constant e) of a value.

Format `LN(Value)`

Where: *Value* = Any positive real number.

Example `LN(12.18)` returns 2.50.
`LN(20.09)` returns 3.00.

See also **EXP, LOG and LOG10 Calc worksheet functions.**

LOG: Calc worksheet function

Purpose Returns the logarithm of a value to the specified base.

Format `LOG(Value [, base])`

Where: *Value* = Any positive real number.
Base = The base of the logarithm. If *Base* is omitted, base 10 is assumed.

Example `LOG(1)` returns 0.
`LOG(10)` returns 1.
`LOG(8, 2)` returns 3.

See also **EXP, LOG10 and LN Calc worksheet functions.**

LOG10: Calc worksheet function

Purpose Returns the base-10 logarithm of a value.

Format `LOG10(Value)`

Where: *Value* = Any positive real number.

Example `LOG10(260)` returns 2.41.
`LOG10(100)` returns 2.

See also **EXP, LOG, and LN Calc worksheet functions.**

LOOKUP: Calc worksheet function

Purpose Searches for a value in one range and returns the contents of the corresponding position in a second range.

Format `LOOKUP (Lookup_value, Lookup_range, Result_range)`

Where: *Lookup_value* = The value for which to search in the first range.
Lookup_range = The first range to search and contains only one row or one column. The range can contain numbers, text or logical values. To search *Lookup_range* correctly, the expression in the range must be placed in ascending order (for example -2, -1, 0, 2 ... A through Z, False, True). The search is not case sensitive.
Result_range = A range of one row or one column that is the same size as the *Lookup_range*.

Example The following examples refer to the **Calc** worksheet cells illustrated below (these cells were hot-linked to a **Data** worksheet, as discussed in Hot-linking Data and Settings worksheet cells to Calc worksheet cells later in this section).
`LOOKUP (0.5, A2:A8, B2:B8)` returns -0.003683852
`LOOKUP (0.4, A2:A8, B2:B8)` returns -0.0023773371 (See remarks).
`LOOKUP (-0.1, A2:A8, B2:B8)` returns #N/A (See remarks).

Figure 6-238
Example LOOKUP Calc worksheet cells

	A	B
1	DrainV(1)	Sourcel(1)
2	0	1.32744E-010
3	0.1000000015	-0.0008447049
4	0.200000003	-0.0016400181
5	0.3000000119	-0.0023773371
6	0.400000006	-0.0030588347
7	0.5	-0.003683852
8	0.6000000238	-0.0042509343
9	0.6999999881	0.0047640027

Remarks If *Lookup_value* does not have an exact match in *Lookup_range*, the largest value that is less than or equal to *Lookup_value* is found, and the corresponding position in *Lookup_range* is returned. When *Lookup_value* does not exist or is smaller than the data in *Lookup_range*, #N/A is returned.

MATCH: Calc worksheet function

Purpose A specified value is compared against values in a range. The position of the matching value in the search is returned.

Format `MATCH(Lookup_value, Lookup_range, Comparison)`

Where: *Lookup_value* = The value against which to compare. It can be a number, text, or logical value or a reference to a cell that contains one of those values.
Lookup_range = Range to search. Contains only one row or one column. The range can contain numbers, text, or logical values.

Comparison = Value representing type of comparison to be made between *Lookup_value* and the values in *Lookup_range*. If you omit *Comparison*, comparison method 1 is assumed.
When *Comparison* is 1, the largest value that is less than or equal to *Lookup_value* is matched. When using this comparison method, the values in *Lookup_range* must be in ascending order (for example, ... -2, -1, 0, 2 ... A through Z, False, True). The search is not case sensitive.
When *Comparison* is 0, the first value that is equal to *Lookup_value* is matched. When using this comparison method, the values in*Lookup_range* can be in any order.
When *Comparison* is -1, the smallest value that is greater than or equal to *Lookup_value* must be in descending order (for example, True, False, Z through A, . . . 2, 1, 0, -1, -2 . . .).

Example The following examples refer to the **Calc** worksheet cells illustrated below (these cells were hot-linked to a **Data** worksheet, as discussed in Hot-linking Data and Settings worksheet cells to Calc worksheet cells).
`MATCH(0.5,A2:A8,1)` returns 6 (the 6th cell relative to cell 2, for example, cell 7).
`MATCH(0.4,A2:A8,1)` returns 4 (the 4th cell relative to cell 2, for example, cell 5).
`MATCH(0.5,A2:A8,0)` returns 6 (because an exact match is found).
`MATCH(0.4,A2:A8,0)` returns #N/A (because an exact match is not found).

Figure 6-239
Example Calc worksheet cells

	A	B
1	DrainV(1)	Sourcel(1)
2	0	1.32744E-010
3	0.1000000015	-0.0008447049
4	0.200000003	-0.0016400181
5	0.3000000119	-0.0023773371
6	0.400000006	-0.0030588347
7	0.5	-0.003683852
8	0.6000000238	-0.0042509343
9	0.6999999881	0.0047640027

Remarks When using comparison method `0` and *Lookup_value* as text, *Lookup_value* can contain wildcard characters. The wildcard characters are `*` (asterisk), which matches any sequence of characters, and `?` (question mark), which matches any single character.

When no match is found for *Lookup_value*, #N/A is returned.

See also **LOOKUP Calc worksheet function.**

MAX: Calc worksheet function

Purpose Returns the largest value in the specified list of numbers.

Format `MAX(`*Value_list*`)`

Where: *Value_list* = A list of as many as 30 numbers separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values.
Error values or text that cannot be translated into numbers return errors.

If a range reference is included in the list, text, logical expression and empty cells in the range are ignored.
If there are no numbers in the list, 0 is returned.

Example `MAX(50, 100, 150, 500, 200)` returns 500.
`MAX(A1:F12)` returns the largest value in this range.

See also **AVERAGE and MIN Calc worksheet functions.**

MIN: Calc worksheet function

Purpose Returns the smallest value in the specified list of numbers.

Format `MIN(Value_list)`

Where: *Value_list* = A list of as many as 30 numbers separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values.
Error values or text that cannot be translated into numbers return errors.
If a range reference is included in the list, text, logical expression, and empty cells in the range are ignored.
If there are no numbers in the list, 0 is returned.

Example `MAX(50, 100, 150, 500, 200)` returns 50.
`MAX(A1:F12)` returns the smallest value in this range.

See also **AVERAGE and MAX Calc worksheet functions.**

MINUTE: Calc worksheet function

Purpose Returns the minutes component of the supplied date/time serial number.

Format `MINUTE(Serial_number)`

Where: *Serial_number* = The time as a serial number. The decimal portion of the number represents time as a fraction of the day.

Example `MINUTE(34506.4)` returns 36.
`MINUTE(34399.825)` returns 48.
`MINUTE(NOW())` returns the present minute of the present hour.

Remarks The result is an integer ranging from 0 to 59.

Needed to extract minutes from the serial number created by the **NOW** function.

See also **DAY, HOUR, MONTH, NOW, SECOND, and YEAR Calc worksheet functions.**

MONTH: Calc worksheet function

Purpose Returns the month component of the supplied date/time serial number or text-formatted date.

Format `MONTH(Serial_number)`

Where: *Serial_number* = The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Example

```
MONTH("06-21-94") returns 6.
MONTH(34626) returns 10.
MONTH(NOW()) returns the present month of the present year.
```

Remarks **MONTH** returns a number ranging from 1 (January) to 12 (December).

Needed to extract the month from the serial number created by the **NOW** function.

See also **DAY, HOUR, MINUTE, NOW, SECOND, and YEAR Calc worksheet functions.**

NOW: Calc worksheet function

Purpose Returns the present date and time as a serial number.

Format

```
NOW()
```

Remarks In a serial number, numbers to the left of the decimal point represent the date, and numbers to the right of the decimal point represent the time. The result of the **NOW** function changes only when a recalculation of the worksheet occurs.

Use the **DAY, HOUR, MINUTE**, **MONTH**, **SECOND**, and **YEAR** functions to extract the information contained in the serial number created by the **NOW** function. These other functions can operate on the **NOW** function in a nested format- for example, HOUR(NOW())- to return results directly.

See also **DAY, HOUR, MINUTE, MONTH, SECOND, and YEAR Calc worksheet functions.**

PI: Calc worksheet function

Purpose Returns the value of pi (π), which is approximately 3.1415926535898 when calculated to 14 significant digits.

Format

```
PI()
```

Where: () = Empty parentheses.

Remarks Although PI does not use arguments, you must supply the empty parentheses to correctly reference this function.

See also **COS, SIN and TAN Calc worksheet function.**

PRODUCT: Calc worksheet function

Purpose Multiplies a list of numbers and returns the result.

Format

```
PRODUCT(Value_list)
```

Where: *Value_list* = A list of as many as 30 numbers, separated by commas.

This list can contain numbers, logical values, text representations of numbers or a reference to a range containing those values.
Error values or text that cannot be translated into numbers return as errors.
If a range reference is included in the list, logical expressions and empty cells in the range are ignored.
All numeric values, including 0, are used in the calculation.

Example `PRODUCT(1, 2, 3, 4)` returns 24.

See also **SUM Calc worksheet function.**

ROUND: Calc worksheet function

Purpose Rounds the given number to the supplied number of decimal places.

Format `ROUND(`*`Value`*`, `*`Precision`*`)`

Where: *`Value`* = Any number.
`Precision` = The number of decimal places to which *`Value`* is rounded.
When a negative precision is used, the digits to the right of the decimal point are dropped and the absolute number of significant digits specified by *`Precision`* are replaced with zeros.
If *`Precision`* is 0, *`Value`* is rounded to the nearest integer.

Example `ROUND(879.278, 2)` returns 879.28.
`ROUND(9899.435, -2)` returns 9900.

SECOND: Calc worksheet function

Purpose Returns the seconds component of the supplied date/time serial number.

Format `SECOND(`*`Serial_number`*`)`

Where: *`Serial_number`* = The time as a serial number (the decimal portion of the number represents time as a fraction of the day).

Example `SECOND(0.259)` returns 58.
`SECOND(34657.904)` returns 46.
`SECOND(NOW())` returns the present second of the present minute.

Remarks Needed to extract seconds from the serial number created by the **NOW** function.

See also **DAY, HOUR, MINUTE, MONTH, NOW, and YEAR Calc worksheet functions.**

SIGN: Calc worksheet function

Purpose Determines the sign of a specified number.

Format `SIGN(`*`Value`*`)`

Where: *`Value`* = Any number.

Example `SIGN(-456)` returns -1.

`SIGN(456)` returns 1.

Remarks **SIGN** returns 1 if the specified number is positive, a -1 if the specified number is negative.

See also **ABS Calc worksheet functions.**

SIN: Calc worksheet function

Purpose Returns the sine of the specified angle.

Format `SIN(Value)`

Where: *Value* = The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by `PI()/180`.

Example `SIN(1.5)` returns 1.76.
`SIN(4.8)` returns -0.996.

See also **ASIN and PI Calc worksheet functions.**

SINH: Calc worksheet function

Purpose Returns the hyperbolic sine of the specified number.

Format `SINH(Value)`

Where: *Value* = Any number.

Example `SINH(1)` returns 1.18.
`SINH(3)` returns 10.02.

See also **ASINH and PI Calc worksheet functions.**

SQRT: Calc worksheet function

Purpose Returns the square root of the specified number.

Format `SQRT(Value)`

Where: *Value* = any positive number. If you specify a negative number, the error #NUM! is returned.

Example `SQRT(25)` returns 5.
`SQRT(160)` returns 12.65.

STDEVP: Calc worksheet function

Purpose Returns the standard deviation of a population based on an entire population of values. The standard deviation of a population represents an average of deviations from the population mean within a list of values.

Format `STDEVP(Value_list)`

Where: *Value_list* = A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example `STDEVP(4.0, 3.0, 3.0, 3.5 2.5 4.0, 3.5)` returns 0.52.

See also **VARP Calc worksheet functions.**

SUM: Calc worksheet function

Purpose Returns the sum of the supplied numbers.

Format `SUM(Value_list)`

Where: *Value_list* = A list of as many as 30 numbers separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values.
Error values or text that cannot be translated into numbers return errors.
If a range reference is included in the list, text, logical expression, and empty cells in the range are ignored.

Example `SUM(1000, 3500, 500)` returns 5000.
`SUM(A10:D10)` returns 6000 if each cell in the range contains 1500.

See also **AVERAGE, PRODUCT, and SUMSQ Calc worksheet functions.**

SUMSQ: Calc worksheet function

Purpose Squares each of the supplied numbers and returns the sum of the squares.

Format `SUMSQ(Value_list)`

Where: *Value_list* = A list of as many as 30 numbers separated by commas. The list can contain numbers, logical values, text representations of numbers, or a reference to a range containing those values.
Error values or text that cannot be translated into numbers return errors.
If a range reference is included in the list, text, logical expression, and empty cells in the range are ignored.

Example `SUMSQ(5, 9, 11)` returns 227.

See also **SUM Calc worksheet function.**

TAN: Calc worksheet function

Purpose Returns the tangent of the specified angle.

Format `TAN(Value)`

Where: *Value* = The angle in radians. If the angle is in degrees, convert the angle to radians by multiplying the angle by `PI()/180`.

Example `TAN(1.5)` returns 14.1.

`TAN(45*PI()/180)` returns 1.

See also **ATAN, PI and TANH Calc worksheet functions.**

TANH: Calc worksheet function

Purpose Returns the hyperbolic tangent of a value.

Format `TANH(Value)`

Where: *Value* = Any number.

Example `TANH(1.5)` returns -0.905.

`TANH(1.1)` returns 0.8.

See also **ATANH, COSH, SINH and TAN Calc worksheet functions.**

VARP: Calc worksheet function

Purpose Returns the variance of a population based on an entire population of values.

Format `VARP(Value_list)`

Where: *Value_list* = A list of as many as 30 numbers, separated by commas. The list can contain numbers or a reference to a range that contains numbers.

Example `VARP(4.0, 3.0, 3.0 3.5, 2.5, 4.0, 3.5)` returns 0.27.

See also **STDEVP Calc worksheet function.**

YEAR: Calc worksheet function

Purpose Returns the year component of the supplied date/time serial number or text-formatted date.

Format `YEAR(Serial_number)`

Where: *Serial_number* = The date as a serial number or as text (for example, 06-21-94 or 21-Jun-94).

Example `YEAR(34328)` returns 1993.

`YEAR("06/21/94")` returns 1994.

`YEAR(NOW())` returns the present year.

Remarks Needed to extract the year from the serial number created by the **NOW** function.

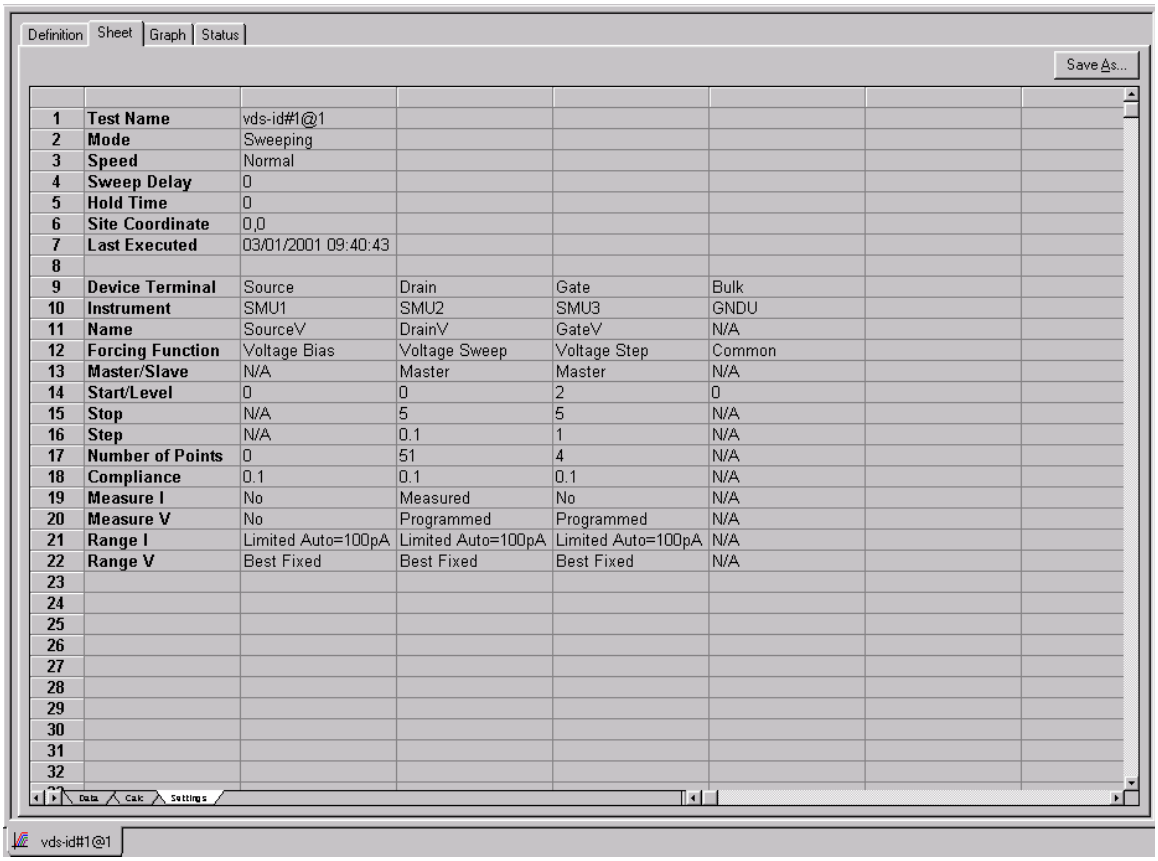
See also **DAY, HOUR, MINUTE, MONTH, NOW, and SECOND Calc worksheet functions.**

Understanding and using the Settings worksheet of a Sheet tab

The **Settings** worksheet records, for the last execution of a test, all test configuration information from the **Definition** tab. All settings values are recorded in Microsoft Excel-compatible format.

Although the **Settings** worksheet is read-only, any of its contents may be hot-linked to the **Calc** worksheet and manipulated there. See Figure 6-240.

Figure 6-240
Settings worksheet



1	**Test Name**	vds-id#1@1				
2	**Mode**	Sweeping				
3	**Speed**	Normal				
4	**Sweep Delay**	0				
5	**Hold Time**	0				
6	**Site Coordinate**	0,0				
7	**Last Executed**	03/01/2001 09:40:43				
8						
9	**Device Terminal**	Source	Drain	Gate	Bulk	
10	**Instrument**	SMU1	SMU2	SMU3	GNDU	
11	**Name**	SourceV	DrainV	GateV	N/A	
12	**Forcing Function**	Voltage Bias	Voltage Sweep	Voltage Step	Common	
13	**Master/Slave**	N/A	Master	Master	N/A	
14	**Start/Level**	0	0	2	0	
15	**Stop**	N/A	5	5	N/A	
16	**Step**	N/A	0.1	1	N/A	
17	**Number of Points**	0	51	4	N/A	
18	**Compliance**	0.1	0.1	0.1	N/A	
19	**Measure I**	No	Measured	No	N/A	
20	**Measure V**	No	Programmed	Programmed	N/A	
21	**Range I**	Limited Auto=100pA	Limited Auto=100pA	Limited Auto=100pA	N/A	
22	**Range V**	Best Fixed	Best Fixed	Best Fixed	N/A	
23						
24						
25						
26						
27						
28						
29						
30						
31						
32						

If the last execution of a test was an **Append** execution, row 7 of the **Settings** worksheet displays the current **Append** worksheet number (**n**) as **(Append = n)**. For example, if the last execution of the **vds-id#1** test generates the **Append4** worksheet, then row 7 (**Last Executed**) displays **(Append=4)** next to the time and date. See Figure 6-241.

Figure 6-241
Settings worksheet after an Append execution

1	**Test Name**	vds-id#1@1
2	**Mode**	Sweeping
3	**Speed**	Normal
4	**Sweep Delay**	0
5	**Hold Time**	0
6	**Site Coordinate**	0,0
7	**Last Executed**	02/20/2001 15:42:08 (Append=4)

NOTE *The **Settings** worksheet entries reflect the last execution of a test. Therefore, if you delete any **Append** worksheets after the last execution of a test, the **Append** worksheet number entry **(Append = n)** does not update until after the next execution.*

Viewing data using the Graph tab

The **Graph** tab allows you to create and export graphs of the test and test-analysis results, which in some cases may be displayed in real time as the test executes. The **Graph** tab provides for flexible plot-data selection, formatting, annotation, and numerical coordinate display (using precision cursors). This subsection covers the following **Graph** tab topics:

- Opening a Graph tab
- Accessing the Graph tab windows
- Defining the data to be graphed
- Defining the axis properties of the graph
- Defining the plot properties of the graph: colors, line patterns, symbols, line widths
- Numerically displaying plot coordinates using cursors
- Viewing plot coordinates and data series properties using the pointing device (mouse)
- Visually reading plot coordinates using cross hairs
- Performing on-graph line fits
- Numerically displaying extracted parameters and other data variables
- Displaying test conditions
- Adding a title, legend, or comment to the graph
- Changing area properties of the graph
- Changing the size of a graph
- Changing the position of a graph
- Identically configuring the graphs resulting from one test executed at multiple sites
- Saving a graph
- Resetting certain graph properties to KITE defaults
- Appending curves from multiple runs on a single graph

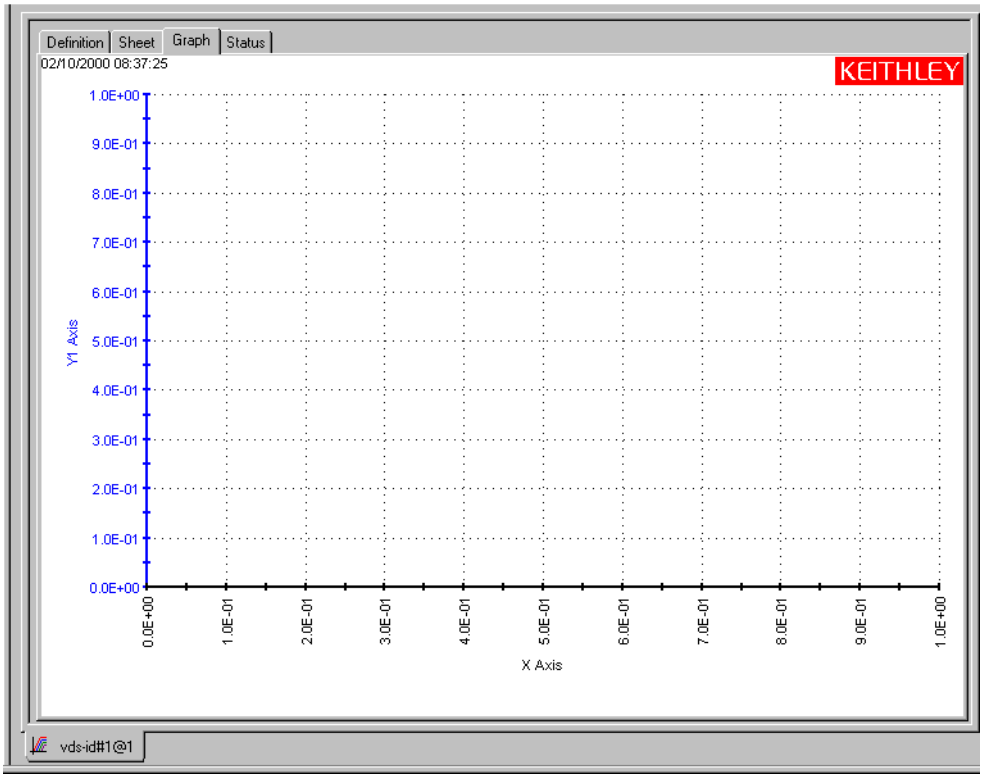
Opening a Graph tab

Open a **Graph** tab as follows:

1. Open the ITM or UTM window for the selected test by double-clicking on the test in the Project Navigator.
2. When the ITM or UTM window opens, click on the displayed **Graph** tab. The **Graph** tab opens.

   Figure 6-242 displays an unconfigured graph for the **"vds-id"** ITM. The time and date at which the data was generated are displayed in the upper left corner. However, the axes are labeled and scaled generically, because no project data has yet been assigned to the axes.

Figure 6-242
Example of an unconfigured graph tab



The vds-id ITM is one of the ITMs that comes installed on your Model 4200-SCS with sample data, including a configured graph (Figure 6-7). The **"vds-id"** ITM has been used for illustration purposes through much of Section 6, including construction of the **u_build** project (Building a completely new Project Plan earlier in this section). The **Definition** tab for the **"vds-id"** ITM is shown in multiple places, including at the beginning of this section. In Figure 6-242, the as-installed configuration has been intentionally removed but will be restored while illustrating **Graph** tab features in the Displaying and analyzing data using the Sheet tab.

Accessing the Graph tab windows

Several **Graph** tab windows control the properties of a graph. You can access these windows two ways, as follows:

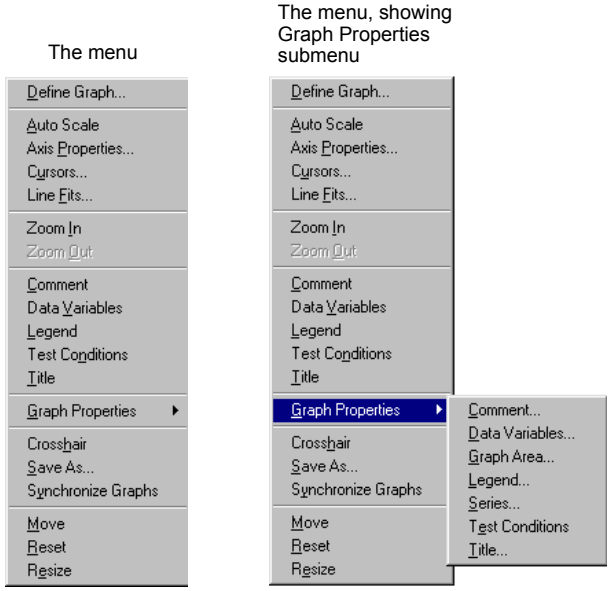
- **By using the Graph Settings menu**: When first defining a graph, you typically access all **Graph** tab windows using the **Graph Settings** menu. Therefore, the next two subsections first tell how to open the **Graph Settings** menu. Then they overview each item on the menu, thereby summarizing the capabilities of the **Graph** tab.
- **By right-clicking on certain graph components**: When certain graph components are already displayed, you can open context-appropriate edit windows by right-clicking on the components. Applicable graph components include titles, legends, comments, numerical coordinates, and values displayed through the **Data Variables** menu item.

Opening the Graph Settings menu

Open the **Graph Settings** menu by either of the following methods:

- **Menu access method I**: Right-click in any blank portion of the **Graph** tab (any place except on a **Graph** tab component). The **Graph Settings** menu appears as a pop-up menu. See Figure 6-243.

Figure 6-243
Graph Settings menu



- **Menu access method II**: In the **Tools** menu of the KITE window, select **Graph Settings**. The menu that appears is identical to the pop-up menu shown in Figure 6-243.

Understanding the Graph Settings menu

Each item of the **Graph Settings** menu is summarized below:

- **Define Graph**: Defines the parameters to be graphed and the axes on which these parameters are to be graphed. For more information, refer to Defining the data to be graphed.
- **Auto Scale**: Automatically scales all axes only once, at a chosen point in time. For more information, refer to Automatically scaling the axes.
- **Axis Properties**: Opens the Axes Properties window, which is the main access point for graph scaling and scale formatting. For more information, refer to Defining the axis properties of the graph.
- **Cursors**: Opens the Cursors window, from which you can select and format cursors that display the precise numerical coordinates of specific points on the plot lines. For more information, refer to Numerically displaying plot coordinates using cursors.
- **Line Fits**: Allows you to directly fit lines to **Graph** tab plots. Up to two fits may be performed on the graph, selected from among the following types: **Linear** (line through two data points), **Regression** (regression line), **Exponential**, **Logarithmic**, and **Tangent**.
- **Zoom In**: Allows you to enlarge and examine a small, selected part of the graph. For more information, refer to Temporarily enlarging a selected area of the graph by zooming.
- **Zoom Out**: Restores a graph to the original or previously zoomed size. For more information, refer to Temporarily enlarging a selected area of the graph by zooming.
- **Comment**: Opens the Comment window, which allows you to add and format a comment. For more information, refer to Adding a comment.

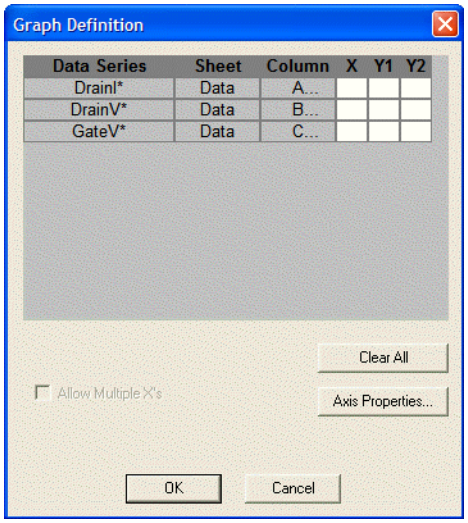
- **Data Variables**: Opens the Data Variables window, from which you can configure the display of up to four data variables, along with the corresponding names (data variables being defined as extracted parameters or other values from the second row of a **Data** or **Calc** worksheet). The **Data Variables** menu item also toggles the data-variable display. For more information about the **Data Variables** item, refer to Numerically displaying extracted parameters and other data variables.
- **Legend**: Toggles the display of an automatically created legend on and off. For more information about legends, refer to Adding a legend.
- **Test Conditions:** Displays the primary test conditions used to obtain the data in the graph. For more information, refer to Displaying test conditions.
- **Title**: Opens the Title window, which allows you to add and format a title. For more information, refer to Adding a title.
- **Graph Properties**
  - **Comment:** Opens the Comment window, which allows you to add and format a comment. Same function as **Comment** in the main menu.
  - **Data Variables**: Opens the Data Variables window, from which you can configure the display of up to four data variables, along with the corresponding names (data variables being defined as extracted parameters or other values from the second row of a **Data** or **Calc** worksheet). Essentially the same as **Data Variables** in the main menu, except that it allows you to open a Data Variables window without toggling the data-variables display.
  - **Graph Area:** Opens the Graph Area menu, which allows you to change the graph foreground and background colors, toggle the time and date display, and make the graph 100% monochrome. For more information, refer to Changing area properties of the graph.
  - **Legend**: Opens the Legend Properties window, which allows you to reformat the font, text or background color, or border of the legend. For more information, refer to Adding a legend.
  - **Series:** Opens the Data Series Properties window, from which you can define color, line pattern, plot symbol, and line width for each plot. For more information, refer to Defining the plot properties of the graph: colors, line patterns, symbols, line widths.
  - **Test Conditions:** Displays the primary test conditions used to obtain the data in the graph. For more information, refer to Displaying test conditions.
  - **Title**: Opens the Title window, which allows you to add and format a title. Same function as **Title** in the main menu.
- **Crosshair**: Toggles the display of a set of cross hairs that can be positioned anywhere on the graph. For more information, refer to Visually reading plot coordinates using cross hairs.
- **Save As**: Opens the Save As window, which allows you to save a graph in bitmap (.bmp) format for use elsewhere, such as in a report. For more information, refer to Saving a graph as a bitmap file.
- **Synchronize Graphs**: For use when the presently open graph is only one of several graphs for the same test, each graph representing the data for a different site. In that case, **Synchronize Graphs** identically configures the graphs for all sites, automatically, using the presently open graph as the master. For more information, refer to Identically configuring the graphs resulting from one test executed at multiple sites.
- **Move**: Toggles between a normal cursor and a crossed-arrow cursor. Moving the crossed-arrow cursor moves the graph, allowing you to relocate it on the **Graph** tab. For more information, refer to Changing the position of a graph.
- **Reset**: Causes colors, graph size, and graph position to be restored to the defaults. For more information, refer to Resetting certain graph properties to KITE defaults.

- **Resize**: Toggles between a normal cursor and a ruler cursor. Moving the ruler cursor expands or contracts the size of the graph. The new size is saved when the graph is saved (By contrast, **Zoom In** affects only the *view* size, which cannot be saved). For more information, refer to Changing the size of a graph.

Defining the data to be graphed

The Graph Definition window is used to define the data to be graphed. Figure 6-244 shows the undefined Graph Definition window for a vds-id ITM.

Figure 6-244
Graph Definition window for a "vds-id" ITM (undefined)



Understanding the table columns in the Graph Definition window

The table columns in the Graph Definition window are used as follows:

- **Data Series**: Lists the names (or other contents[9]) of every first-row cell of the **Data** and **Calc** worksheets. If you have generated **Append** worksheets[10] for the test, the Data Series column also lists the names of every first-row cell in every **Append** worksheet. However, when multiple first-row cells name the same parameter (because multiple sets of data exist under that name) the following applies:
  - The name of the parameter is listed only once under **Data Series**, because it corresponds to a family of curves.
  - Asterisks (*) appear next to all parameter names listed under **Data Series.**
- **Sheet**: Indicates whether the data comes from the **Data** worksheet, the **Calc** worksheet, or a specific **Append** worksheet.
- **Column**: Lists the parameter's **Data, Calc,** or **Append** worksheet column label (A, B, C, etc)..
- **X, Y1,** and **Y2:** Are the axes of the graph, as follows:
  - **X** is the X axis.
  - **Y1** is the Y axis on the left side of the graph.

9. KITE assumes that first-row cells contain variable names. However, a first-row **Calc** worksheet cell is allowed to contain a number, and KITE displays such a number under **Data Series.** Therefore, in general, avoid placing numbers (or any unwanted plot parameter names) in the first row of a **Calc** worksheet**.**
10. For more information about generation and use of **Append** worksheets, refer to "Append execution of tests, test sequences, and Project Plans," "Understanding and using Append worksheets of a Sheet tab," and "Appending curves from multiple runs on a single graph."

- **Y2** is the Y axis on the right side of the graph.

NOTE *The scale and label of the **Y2** axis are allowed to be different from the scale and label of the **Y1** axis.*

The cells under the **X, Y1,** and **Y2** may be selected and deselected by clicking the boxes.

- If you select a cell under **X**, the corresponding **Data Series** parameter is plotted on the X axis. KITE can plot multiple parameters on the **X** axes when the test does not define a family of curves (see Allow Multiple X's).
- Likewise, if you select a cell under **Y1** or **Y2**, the corresponding **Data Series** parameter is plotted on the **Y1** axis or the **Y2** axis, respectively. KITE can plot multiple parameters on the **Y1** and **Y2** axes.

Understanding the buttons in the Graph Definition window

The buttons of the Graph Definition window are used as follows:

- **Clear All:** A click of the **Clear All** button clears all selections under columns **X**, **Y1** and **Y2**.

NOTE *If you click the **Clear All** button by mistake, click the **Cancel** button to exit the Graph Definition window without making any changes.*

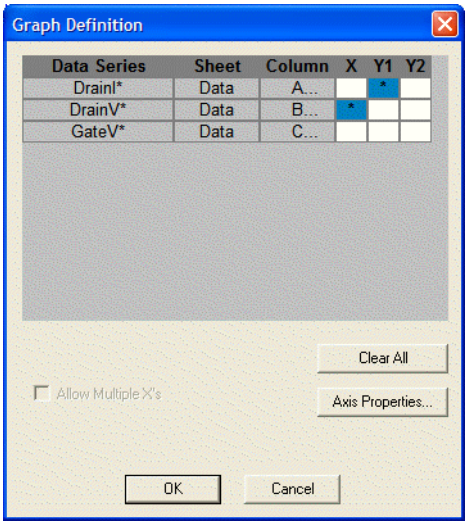
- **Axis Properties**: a click of the **Axis Properties** button opens the Axis Properties window. You can also open the Axis Properties window by selecting **Axis Properties** in the **Graph Settings** menu. Before using the Axis Properties window, refer to the next subsection, Defining the axis properties of the graph.

Opening and using the Graph Definition Window

Open and use the Graph Definition window, as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Define Graph.** The Graph Definition window opens.
3. Using the Graph Definition window, indicate which parameters are to be plotted and assign them to appropriate axes by selecting the appropriate **X**, **Y1**, and **Y2** cells.

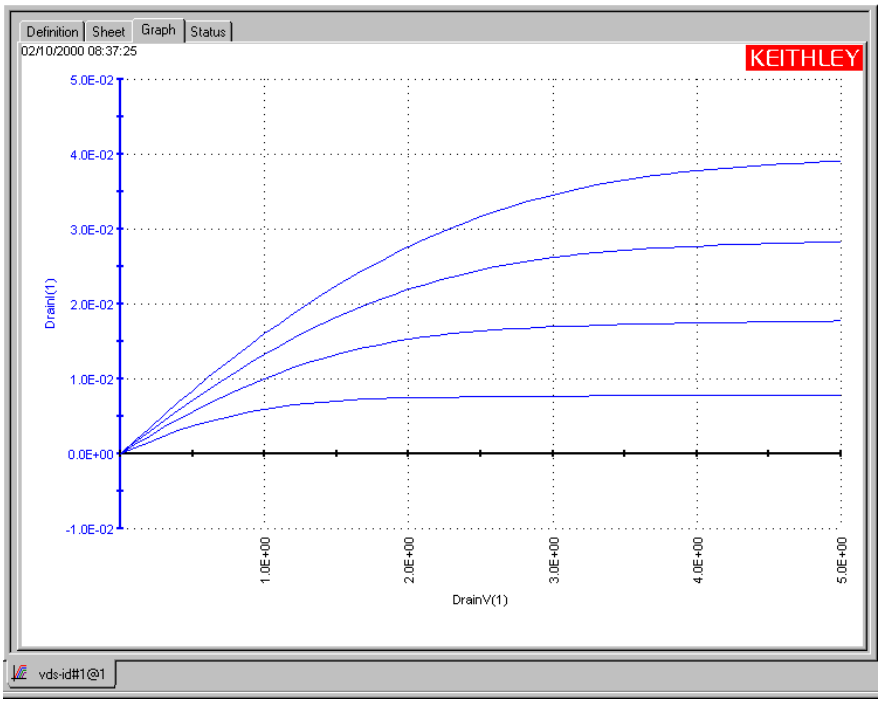
Figure 6-245
Configured Graph Definition window for a "vds-id" ITM



4. Click **OK**. The graph now displays plots of the selected parameters.

   In Figure 6-246, the vds-id graph now displays scaled axes and a series of four plots, reflecting on the selections shown in Figure 6-245. The family of curves corresponds to four sets of data generated by drain-voltage sweeps at four different gate voltages.

Figure 6-246
"vds-id" graph after configuring its Graph Definition window

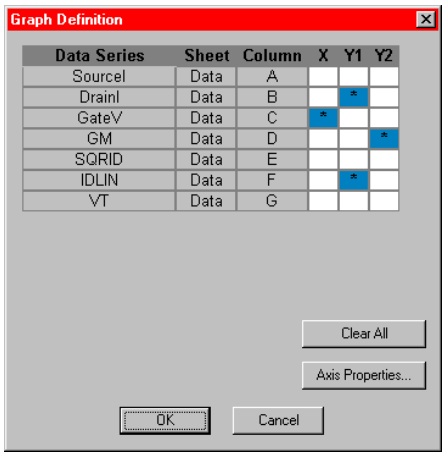


The axis labels shown in Figure 6-246 are not yet optimally named. KITE inserted the default **Data** sheet column labels for sweep #1 of the data series. However, the axis labels will be renamed in the next subsection, Defining the axis properties of the graph.

Figure 6-247 shows the Graph Definition window for an enhanced vgs-id ITM. This is a somewhat more complex graph definition than shown in Figure 6-245, because:

- It specifies two parameters to be plotted on the **Y1** axis: a raw data parameter and a calculated parameter.
- It specifies one parameter to be plotted on the **Y2** axis: a calculated parameter.

Figure 6-247
Configured Graph Definition window for a "vgs-id" ITM



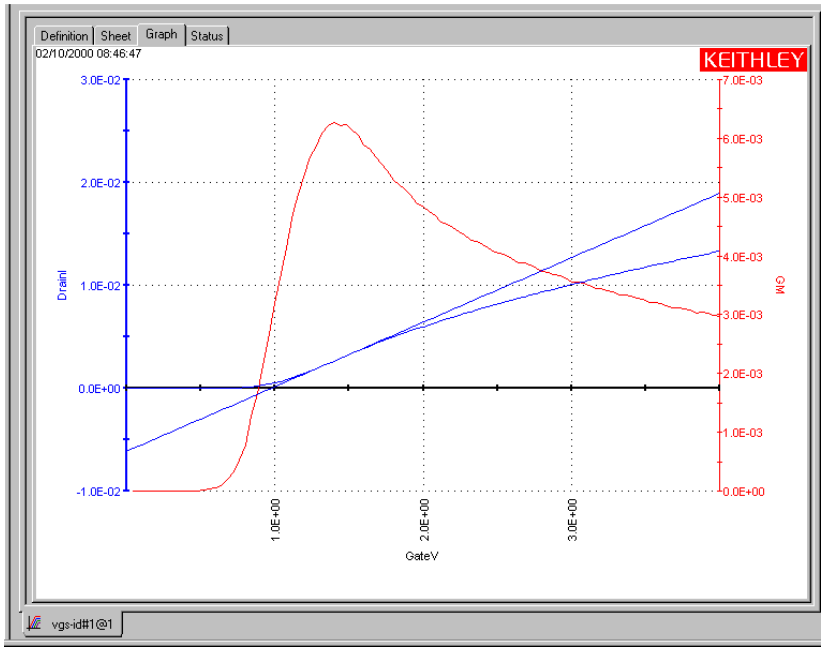
However, note that there are no asterisks (*) next to the entries in the Data Series column; this graph definition is for a single set of data and data-derived (calculated) parameters.

Figure 6-248 shows the default vgs-id graph that results from the graph definition of Figure 6-247. Note that the **Y1** axis and its two plots are identically color-coded with blue. The **Y2** axis and its single plot are identically color-coded with red.

NOTE *Once the graph is defined for a test (and configured as described subsequently) the graph displays the selected results of the present **Data** or **Calc** worksheets or both. Each subsequent run updates the graph as follows:*

- *After executing the test in **Run** mode, the new curves replace the existing curves.*
- *After executing tests in **Append** mode, the new curves append (layer on top of) the existing curves, up to a preset limit. For details on the **Append** mode, refer to Append execution of tests, test sequences, and Project Plans and Appending curves from multiple runs on a single graph*

Figure 6-248
"vgs-id" graph after configuring its Graph Definition window

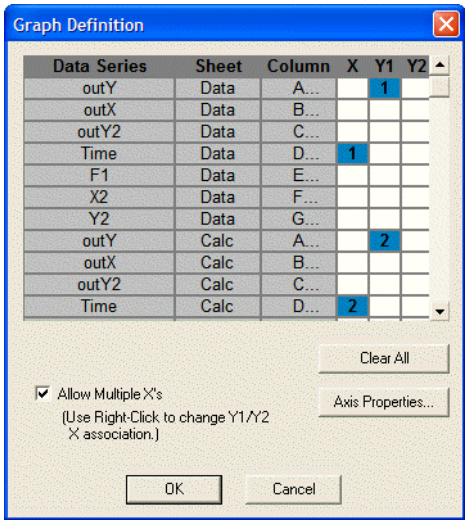


Allow Multiple X's

When the test is not defining a family of curves, the **Allow Multiple X's** option can be selected. Figure 6-249 shows an example of multiple X's selected for a qualified test. As indicated by the selections, there is an X axis for Time(1) and another one for Time(2). The data for outY(1) corresponds to Time(1), and the data for outY(2) corresponds to Time(2).

If cells under Y2 were selected, right-clicking on the definition window will toggle the Y1/Y2 X association.

Figure 6-249
Example Graph Definition using multiple X's

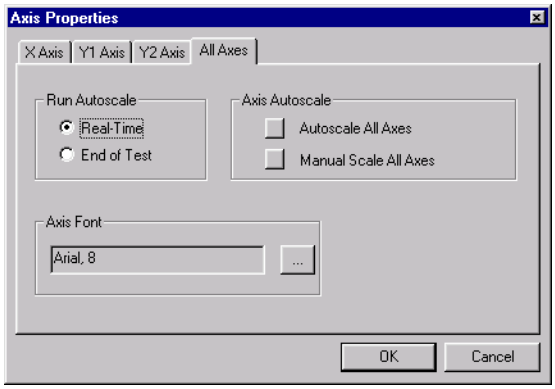


Defining the axis properties of the graph

The Axis Properties window is the main access point for graph scaling and scale formatting. Open the Axis Properties window as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Axis Properties.** The Axis Properties window opens. Figure 6-250 shows an Axis Properties window.

Figure 6-250
Axis Properties window



The four tabs of the Axis Properties window are used as follows:

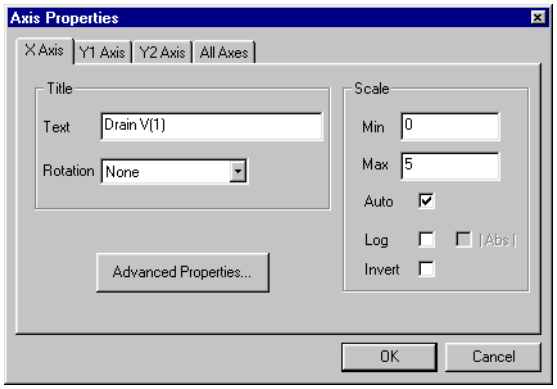
- **X Axis**: Controls properties of the horizontal axis.
- **Y1 Axis**: Controls properties of the left vertical axis.
- **Y2 Axis**: Controls properties of the right vertical axis.
- **All Axes**: Controls certain properties that are common to all axes.

Naming the axes

When you first open a **Graph** tab and specify the variables to be plotted, the default axis names (titles) are the same as the **Data** sheet column headings in row 1. If there are multiple sets of data with the same parameter names, the default axis names are the column headings for set #1, for example, **DrainI(1)** and **DrainV(1).**[11] To rename an axis, do the following:

1. In the **Graph** tab Axis Properties window, open the **X Axis**, **Y1 Axis**, or **Y2 Axis** tab, as appropriate, by clicking on the tab label. The tab opens. Figure 6-251 shows the **X Axis** tab.

Figure 6-251
X Axis tab



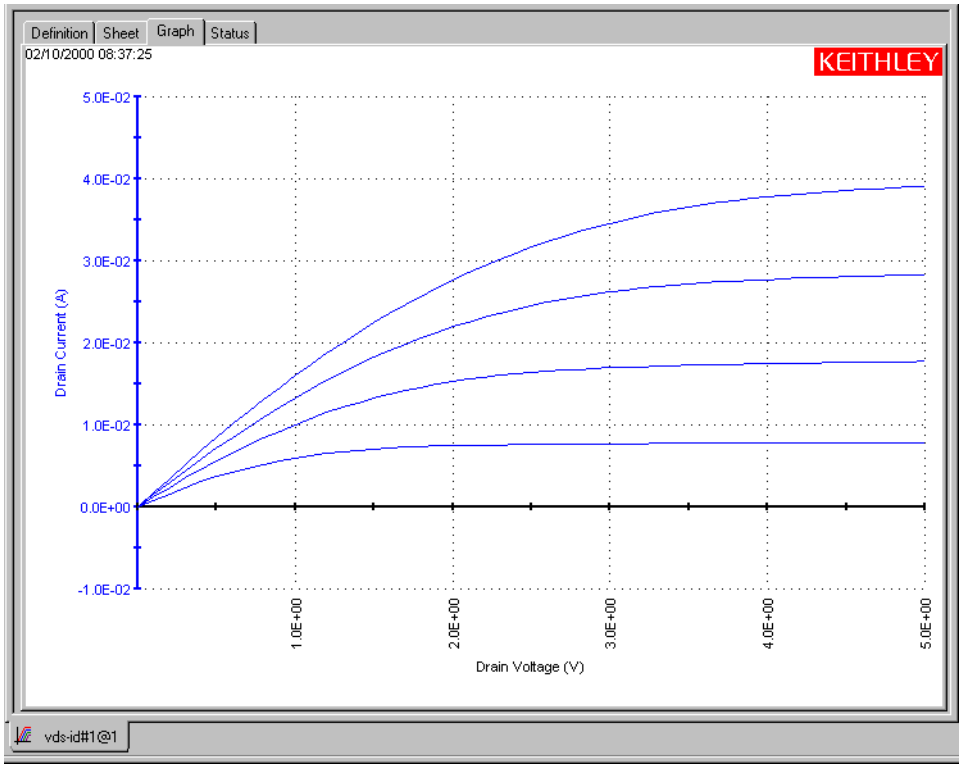
NOTE *Alternatively, right-clicking on an axis displays the corresponding axis tab.*

2. In the **Text** edit box, type the new name (title) for the axis, preferably including measurement units. For example, for the **X** axis of the vds-id graph you might type **Drain Voltage (V)**.
3. In the **Rotation** combo box, specify a different orientation for the axis name (title) if the default orientation is not desired. The angle is specified relative to the **X** axis. By default, axis names are parallel to and facing the axes: 0 degrees for **X**, 90 degrees for **Y1**, and 270 degrees for **Y2**.
4. Name the other axes by repeating steps 1 through 3.

11. The Graph axes of some library ITMs may already be labeled more appropriately. For example, a **DrainV(1)** column label in the **Data** worksheet may have already been converted to a **Drain Voltage (V)** axis label in the **Graph** tab.

5. Click **OK**. The graph reflects the new axis names. Figure 6-252 shows the results of renaming the vds-id graph axes, in place of the KITE default names (as shown in Figure 6-251).

Figure 6-252
Renamed "vds-id" graph axes



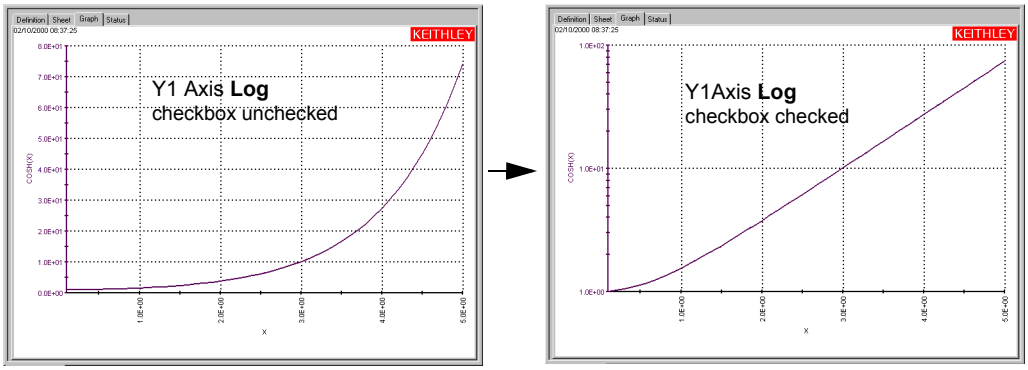
Specifying a logarithmic scale for an axis

In some cases, particularly when a test parameter spans several decades, you may desire to use a logarithmic axis. To change a linear axis to a logarithmic axis, do the following:

1. In the **Graph** tab Axis Properties window, open the **X Axis**, **Y1 Axis**, or **Y2 Axis** tab for the axis to be changed. The tab opens.
2. In the selected axis tab, click the **Log** checkbox. The **|Abs|** checkbox becomes enabled.
3. If all values on the selected axis are all positive, skip to step 5.
4. If any values on the selected axis are negative (and therefore, mathematically, cannot be converted to logarithms), do one of the following:
   - If values on the selected axis are all negative:
   a. Click the **|Abs|** (absolute value) checkbox. This converts all negative values to positive values before plotting them on a log scale.
   b. Skip to step 5.
   - If values on the selected axis are both positive and negative, click the **|Abs|** (absolute value) checkbox. Be aware that you will observe two curves that typically do not connect smoothly: one for the intrinsically positive values and another for the intrinsically negative values (to graph only the positive values or only the **|Abs|** converted absolute values refer to Manually scaling the axes).
5. Click **OK**. The tab-selected axis is now scaled logarithmically.

Figure 6-253 illustrates a linear-to-log scale modification.

Figure 6-253
Example of a linear-to-log scale modification



NOTE *Autoscaling, discussed subsequently under "Automatically scaling the axes," works with both linear and logarithmic scales. However, autoscaling may change the way a logarithmic scale is displayed.*

Plotting on an inverted scale

You can invert the direction in which a variable is plotted. In other words, you can configure an **X** axis such that the values increase from right to left, instead of from left to right. Likewise, you can configure a **Y1** or **Y2** axis such that the values increase from top to bottom, instead of from bottom to top. To invert the scale of an axis, do the following:

1. In the **Graph** tab Axis Properties window, open the **X Axis**, **Y1 Axis**, or **Y2 Axis** tab for the axis to be changed. The tab opens.
2. In the selected axis tab, click the **Invert** checkbox.
3. Click **OK**. The tab-selected axis is now inverted. Figure 6-254 illustrates an **X** axis inversion. Figure 6-255 illustrates a **Y1** axis inversion (the effects are similar for a **Y2** axis inversion).

NOTE *You must use the Move menu selection to properly position the graph after an axis inversion. For instructions, refer to Changing the position of a graph.*

Figure 6-254
Example of an X axis inversion

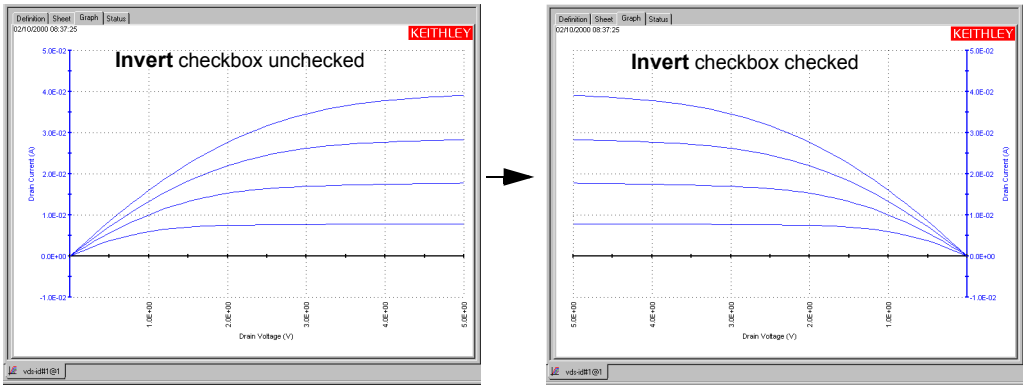
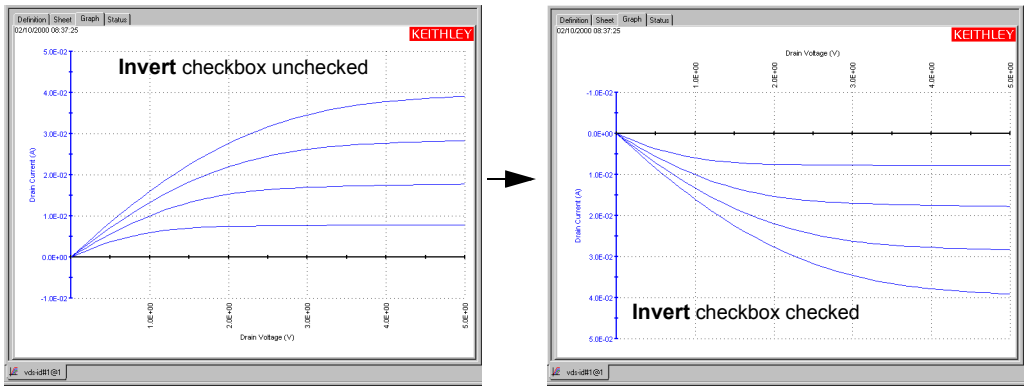


Figure 6-255
Example of a Y1 Axis inversion



Automatically scaling the axes

The **Autoscale** feature of the **Graph** tab optimizes the scale of an axis to show all of the data, based on the largest value to be plotted. For a new graph, all axes are autoscaled by default.

Individually setting an axis to autoscale every time the test executes

To set individual **X** axis, **Y1** axis, or **Y2** axis to **Autoscale,** do the following:

1. In the **Graph** tab Axis Properties window, open the **X Axis, Y1 Axis**, or **Y2 Axis** tab by clicking on it.
2. Click the **Auto** checkbox.
3. In the Axis Properties window, open the **All Axes** tab by clicking on it.
4. Select either the **Real Time** radio button or the **End Of Test** radio button. The **Real Time** and **End Of Test** radio buttons control when axis autoscaling takes place, as follows:
   - When the **Real Time** radio button is checked, the axes that are set to **Autoscale** are autoscaled with *every* data point that is acquired.

NOTE *When you specify **Real Time**, you can watch KITE plot an ITM graph in real time—during test execution. This option applies to all raw-data parameters and some calculated parameters (for **Formulator** calculated parameters, refer to "Real-time functions, operators, and formulas," and Post-test-only functions and formulas). If a real-time plot spans several decades, then autoscaling allows you to temporarily observe even the lowest-range data at meaningful scales.*

*The **Real Time** option does not apply to UTM data and calculations. A UTM **Data** worksheet, and therefore a UTM **Graph** tab, does not update until the test is finished.*

   - When the **End Of Test** radio button is checked, the axes that are set to **Autoscale** are autoscaled only after the test is complete.

NOTE *If the **Auto** checkbox is unchecked in an **X Axis, Y1 Axis,** or **Y2 Axis** tab, the corresponding axis is unaffected by the status of the **Real-Time** and **End-Of-Test** radio buttons.*

Simultaneously setting all axes to autoscale every time the test executes

To simultaneously set all of the axes (**X**, **Y1**, and **Y2**) to **Autoscale**, do the following:

1. In the **Graph** tab Axis Properties window, open the **All Axes** tab by clicking on it.
2. Under **Axis Autoscale**, click the **Autoscale All Axes** button.
3. Select either the **Real Time** checkbox or the **End Of Test** checkbox. The **Real Time** checkbox and the **End Of Tes**t boxes control when axis autoscaling takes place, as follows:
   - When the **Real Time** box is checked, ITM graph axes autoscale with *every* data point that is acquired.

NOTE *When you specify **Real Time**, you can watch KITE plot an ITM graph in real time, during test execution. This option applies to all raw-data parameters and some calculated parameters (for **Formulator** calculated parameters, refer to Real-time functions, operators, and formulas, and Post-test-only functions and formulas). If a real-time plot spans several decades, then autoscaling allows you to temporarily observe even the lowest-range data at meaningful scales.*

*The **Real Time** option does not apply to UTM data and calculations. See Enabling real time plotting for UTMs to plot UTM data in real time.*

   - When the **End Of Test** box is checked, the axes autoscale only after the test is complete.

Simultaneously setting all axes to autoscale only once

To autoscale all axes only once, at a chosen point in time, do the following:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Auto Scale.** The following occurs:
   - All three axes are autoscaled.
   - Autoscaled **Min** and **Max** settings replace any manually scaled **Min** and **Max** settings in the **X Axis, Y1Axis**, or **Y2 Axis** tabs (**Min** and **Max** are settings that are selected according to Manually scaling the axes).
   - The **Auto** checkboxes of the **X Axis, Y1 Axis**, and **Y2 Axis** tabs are restored to their pre-autoscale status. For example, if the **Auto** checkbox of the **X Axis** tab was unchecked before you selected **Graph Settings > Auto Scale,** it is also unchecked thereafter.

Manually scaling the axes

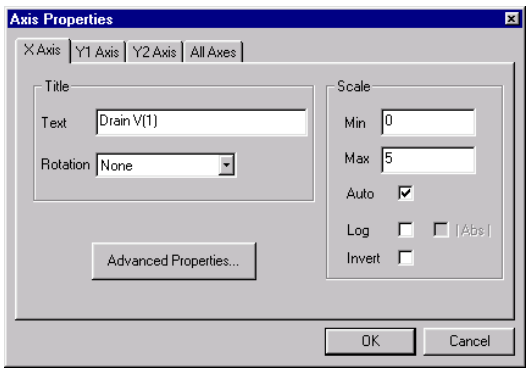
When an axis is manually scaled, KITE holds the scales of the axis constant, both during and after a test, based on the **Max** and **Min** values set for the axis. You manually scale each axis individually. Therefore, it is possible to manually scale some axes and automatically scale other axes.

Manually scaling individual axes

Scale an individual **X**, **Y1**, or **Y2** axis as follows:

1. In the **Graph** tab Axis Properties window, click on the label of the **X Axis, Y1 Axis**, or **Y2 Axis** tab, as appropriate. The tab opens. Figure 6-256 shows the **X Axis** tab.

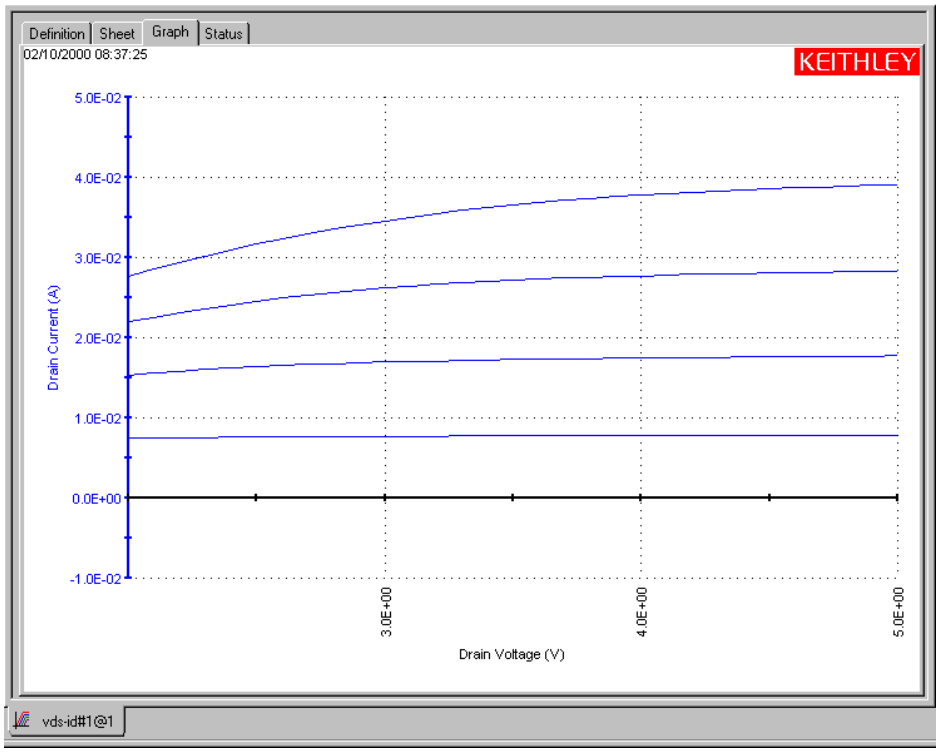
Figure 6-256
X Axis tab



2. On the **X Axis**, **Y1 Axis,** or **Y2 Axis** tab, do the following:
   a. If the **Auto** checkbox is checked, click it to uncheck it. If the **Auto** checkbox is not checked, leave it unchecked.
   b. In the **Min** edit box, type the *minimum* value that you want to be plotted and labeled on the axis.

      For example, to display only a part of the graph, you might wish to set the manually scaled **Min** value to be larger than the smallest data value. Figure 6-257 shows the result of setting **Min** to **2** on the **"vds-id" X Axis** tab (and then clicking **OK**).

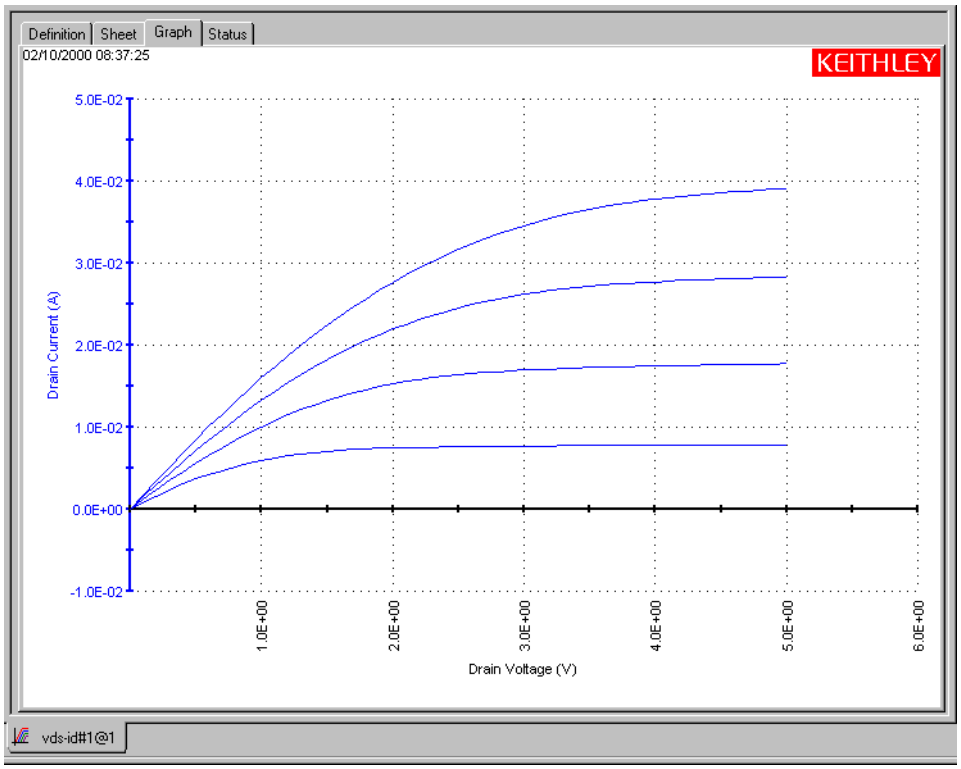
Figure 6-257
"Vds-id" graph after setting the X Axis tab "Min" value to "2"



c. In the **Max** edit box, type the *maximum* value that you want to be plotted and labeled on the axis.

   For example, you might wish to set the manually scaled **Max** value to be larger than the largest data value to allow for text at the right side of the graph. Figure 6-258 shows the result of setting **Max** to **6** on the **"vds-id" X Axis** tab (and then clicking **OK**).

Figure 6-258
"Vds-id" graph after setting the X Axis tab "Max" value to "6"



3. Repeat step 2 for other axes that you wish to manually scale.
4. Click **OK**. The graph displays the new scaling.

Simultaneously changing all axes from autoscaling to manual scaling

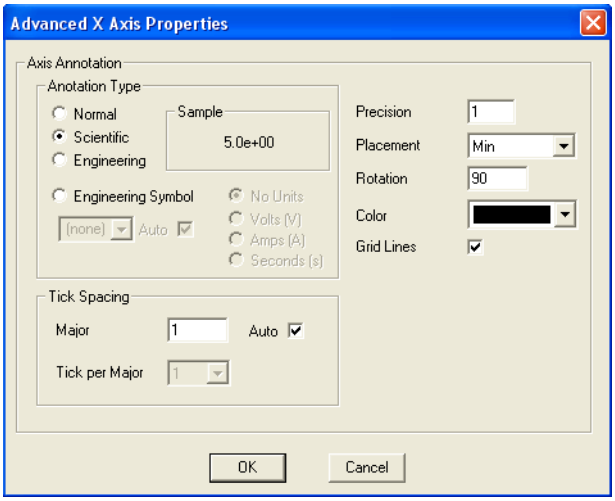
To simultaneously change all of the axes from autoscaling to manual scaling, click the **Manually Scale All Axes** button on the **All Axes** tab of the **Graph** tab Axis Properties window**.** The **Max** and **Min** settings on the **X Axis, Y1 Axis**, and **Y2 Axis** tabs are then fixed at the values that were optimized by the autoscaling operation (any **Max** and **Min** values that were manually set *before* the autoscaling operation were automatically replaced by optimized values during the autoscaling operation).

Customizing axis locations and formats

In the **Graph** tab Axis Properties window, clicking the **Advanced Settings** button on an **X Axis**, **Y Axis**, or **Y2 Axis** tab opens an Advanced X Axis Properties, Advanced Y1 Axis Properties, or Advanced Y2 Axis Properties window. The three Advanced Axis Properties windows (one for each axis) provide additional controls to customize the axes.

Figure 6-259 shows the default Advanced X Axis Properties window.

Figure 6-259
Default Advanced X Axis Properties window



The items for the **Advanced X Axis Properties** are explained as follows:

- **Annotation Type**:
  - **Normal**: When selected (•), specifies the axis labels are in simple decimal notation (for example, **30.0**).
  - **Engineering**: When selected (•), specifies that the axis labels are in engineering notation (for example, **300E-3** instead of **0.30**).
  - **Scientific**: When selected (•), specifies that the axis labels are in scientific notation (for example, **3.0E+01** instead of **30.0**).
  - **Engineering Symbol**: When selected (•), the labels will include the engineering symbol. With **Auto** selected (√), the symbol will be added automatically. With **Auto** disabled, the adjacent drop-down menu becomes active for manual selection of a symbol (for example, **30.0 m**).
    - **No Units**, **Volts (V)**, **Amps (A)** or **Seconds (s)**: With Engineering Symbol selected, units (or no units) can be selected (for example **30.0 mA**).
- **Precision edit box**: Specifies the number of decimal points in the labels.

- **Placement combo box**: Specifies where the **X** axis labels are placed relative to the top and bottom of the graph and where **Y1** axis and **Y2** axis labels are placed relative to the right and left sides of the plot. For all axes, the default **Auto** selection allows KITE to decide where the labels are placed. Figure 6-260 and Figure 6-261 describe the **Max, Min**, and **Origin** selections for **X** and **Y1** axes (the same principle applies to the **Y2** axis). The **Origin** selection is designed for a bipolar axis, having both positive and negative scale values. If an axis is not bipolar, the **Origin** selection is the same as the **Min** selection).

Figure 6-260
X axis placement options

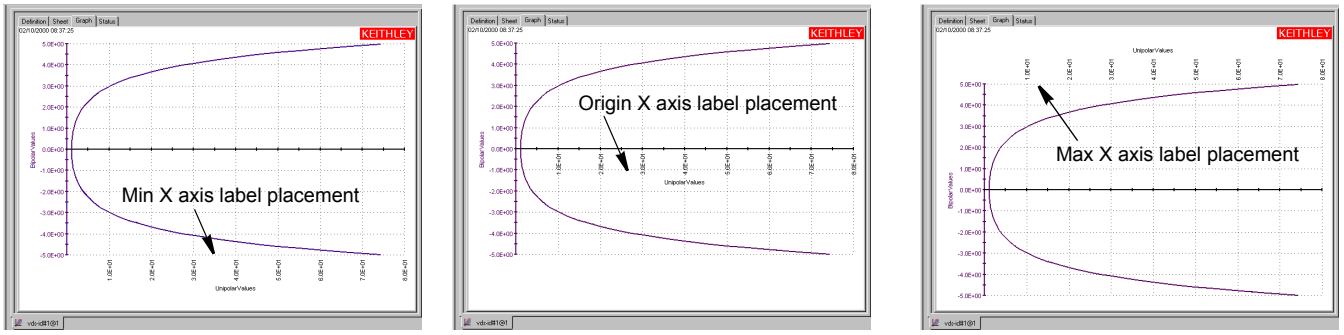
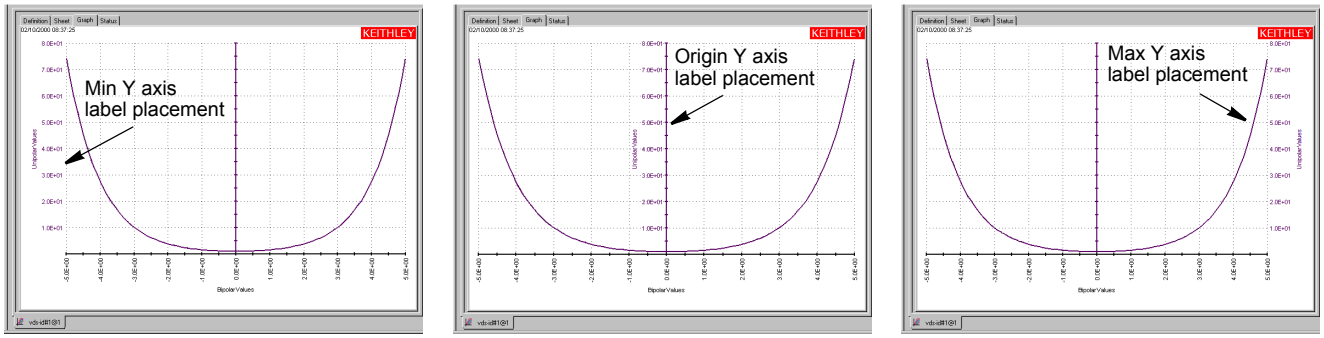


Figure 6-261
Y1 axis placement options



- **Rotation edit box**: Specifies the alignment of the labels of the tab specified axis. All angles are specified relative to the **X** axis. The default rotations place the labels perpendicular to the axes, as follows:
  - The default **X** axis label rotation is 90 degrees.
  - The default **Y1** axis label rotation is 0 degrees.
  - The default **Y2** axis label rotation is 0 degrees.
- **Color combo box**: Specifies the color of the labels of the tab specified axis. **Black** is the default for the **X** axis, **Blue** is the default for the **Y1** axis, and **Red** is the default for the **Y2** axis.
- **Grid Lines checkbox**: Specifies whether the graph is to have grid lines at the major tick marks of the tab specified axis. The **Grid Lines** checkbox is checked by default.

- **Tick Spacing:**
  - **Auto checkbox**: Specifies whether KITE automatically calculates and implements an appropriate **Major** tick spacing for the tab specified axis. The **Auto** checkbox is checked by default.

- **Major tick edit box**: Specifies the spacings between the individual labels on the tab specified axis and between the individual tick marks and grid lines, in terms of actual plot units. If the **Auto** checkbox is not checked, you can specify the tick spacing manually. For example if the **X** axis range is 5 V, you might specify **0.2** to space the labels and major tick marks 0.2V apart.
- **Tick per Major combo box**: Specifies the number of ticks to be placed between the major ticks on the tab specified axis. If the Auto checkbox is checked, the Tick per Major combo box is automatically set to 1. Otherwise, you can manually set the Tick per Major value from 1 to 4.

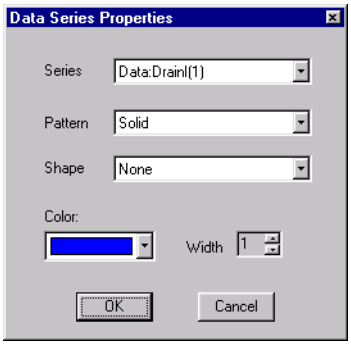
NOTE *If you autoscale all axes simultaneously by selecting **Auto Scale** in the **Graph Settings** menu, the **Major** tick is set to **Auto** momentarily during the scale update, and the **Major** tick setting changes appropriately at the completion of the autoscale operation. However, the manual **Tick per Major** setting is retained at the completion of the autoscale operation.*

Defining the plot properties of the graph: colors, line patterns, symbols, line widths

You define color, line pattern, plot symbol, and line width for a plot through the Data Series Properties window. Open it as follows:

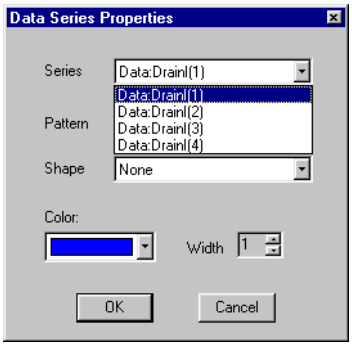
1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Graph Properties > Series.** The Data Series Properties window appears. See Figure 6-262.

Figure 6-262
Data Series Properties window for the "vds-id" ITM



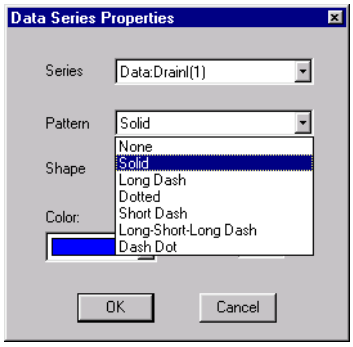
3. In the Data Series Properties window **Series** combo box, select the plot series for which you desire to set properties (every **Y1** and **Y2** parameter for every series is listed). For the vds-id graph there are four selections, as shown in Figure 6-263.

Figure 6-263
Example of Series selections



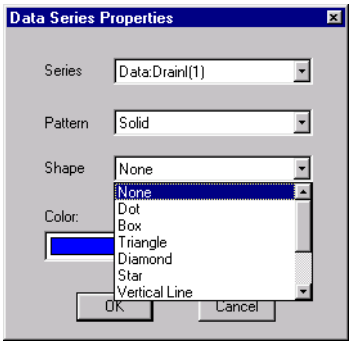
4. In the **Pattern** combo box, select a special line pattern, if desired, for the plot line. Figure 6-264 shows the available line patterns.

Figure 6-264
Line-pattern selections



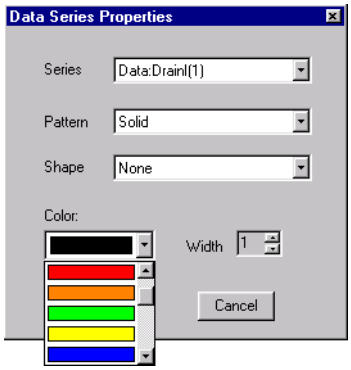
5. In the **Shape** combo box, select a special plot symbol, if desired, for the plot line. Figure 6-265 shows most of the available plot symbols.

Figure 6-265
Most of the plot-symbol selections



6. In the **Color** combo box, select a special line color for the plot line if desired. Figure 6-266 shows some of the available non-black line colors.

Figure 6-266
Some of the plot-color selections



7. In the **Width** combo box, select the line width for the plot line if desired. The line width is variable between 1 and 9 screen pixels (the default is one pixel).
8. As needed, repeat steps 3 through 7 for each plot on the graph.
9. Click **OK**. The graph reflects the new plot line definitions.

   See the following examples:

   - Figure 6-267 shows the vds-id plot lines with variable line patterns, set through the **Pattern** combo box.
   - Figure 6-268 shows the vds-id plot lines with plot symbols, set through the **Shape** combo box.
   - Figure 6-269 shows the vds-id plot lines colored through the **Color** combo box.
   - Figure 6-270 shows the colored vds-id plot lines changed to 2-pixel width through the **Width** combo box.

Figure 6-267
"vds-id" plot lines with variable line patterns, set through the Pattern combo box

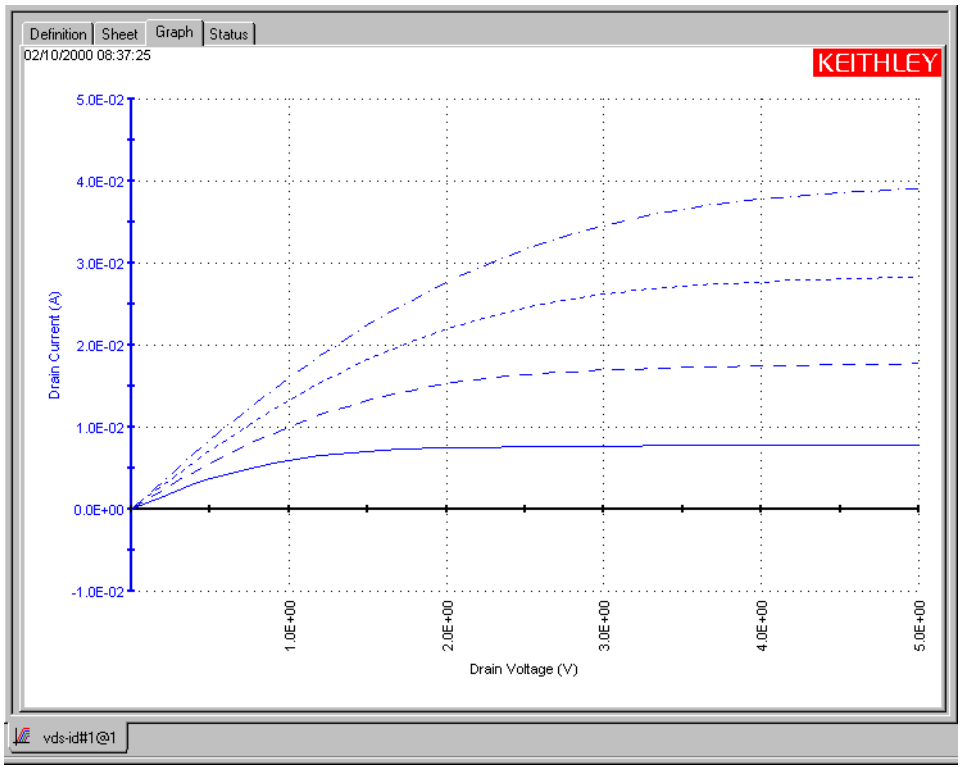


Figure 6-268
"vds-id" plot lines with plot symbols, set through the Shape combo box

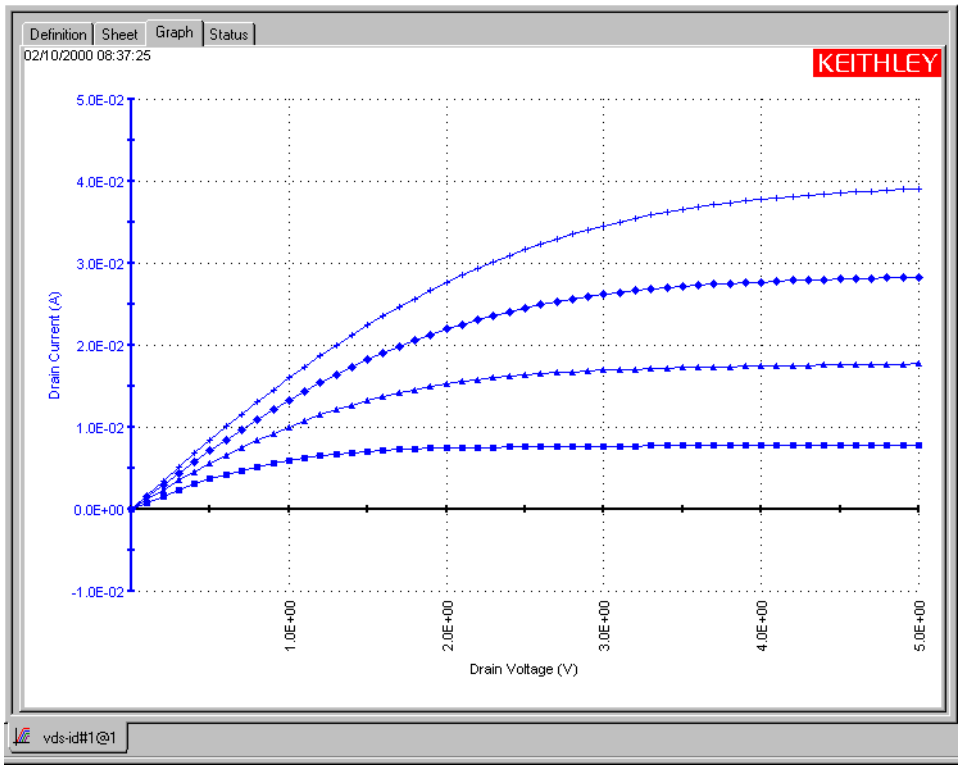


Figure 6-269
"vds-id" plot lines colored through the Color combo box

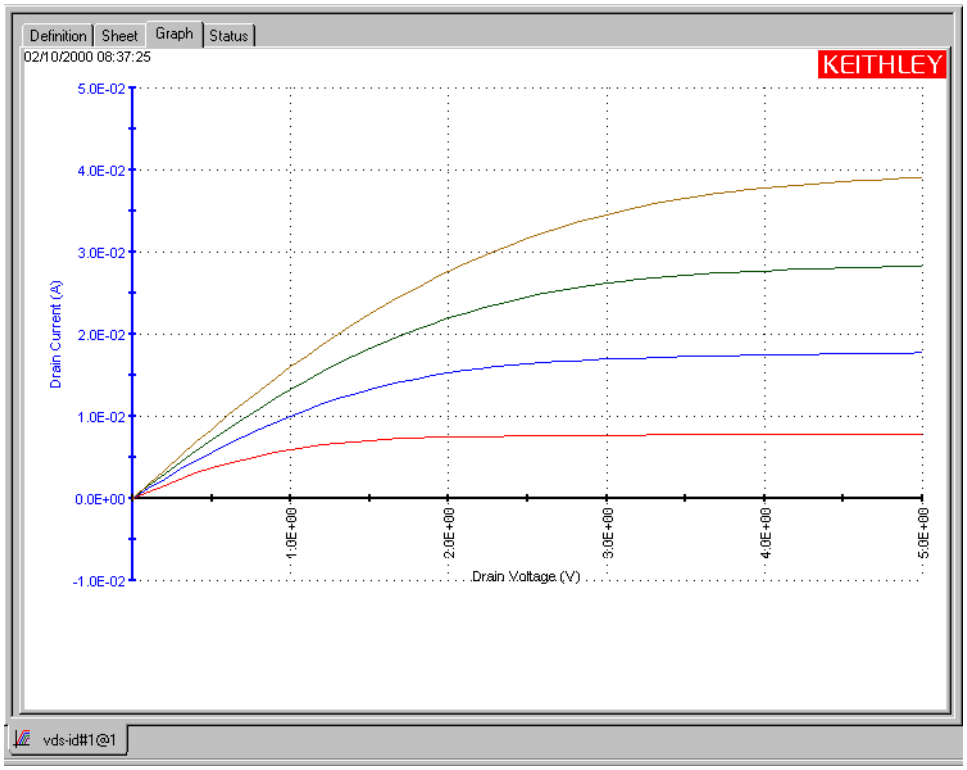
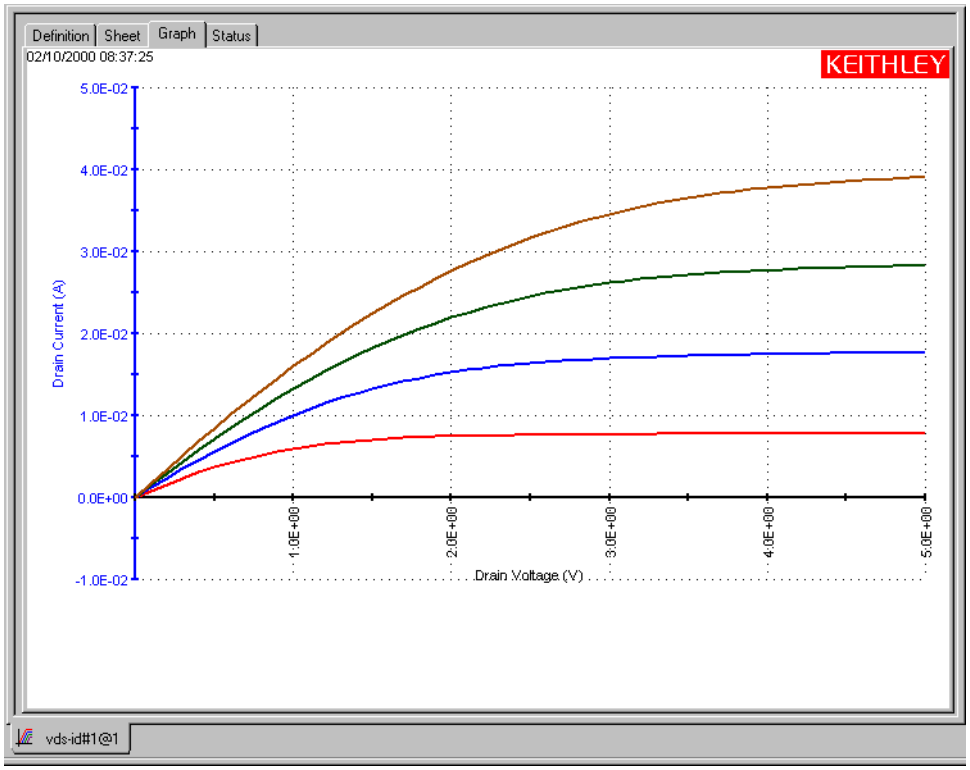


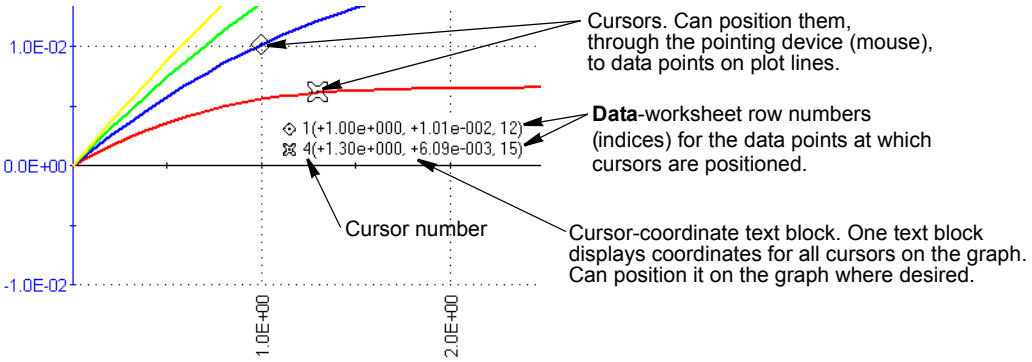
Figure 6-270
Colored "vds-id" plot lines widened to a 2-pixel width through the Width combo box



Numerically displaying plot coordinates using cursors

You can display the precise numerical coordinates of a specific data point on a plot using a **Graph** tab cursor. When you move a cursor, it precisely tracks the plot to which it is attached. Wherever you stop a cursor, a displayed text block indicates the precise X,Y coordinates of the stopping point. See Figure 6-271.

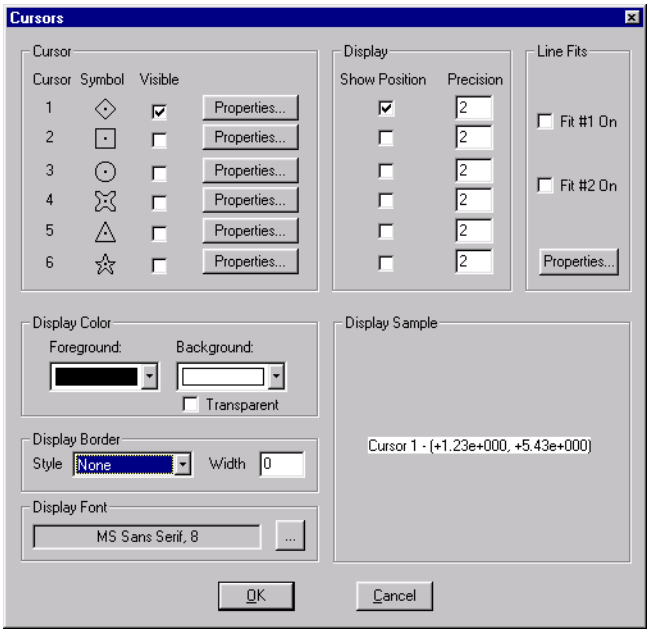
Figure 6-271
Cursor illustration



To work with cursors, first open the **Graph** tab Cursors window, as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Cursors.** The Cursors window opens. See Figure 6-272.

Figure 6-272
Cursors window



Specifying and configuring the cursors

Specify the cursors as follows:

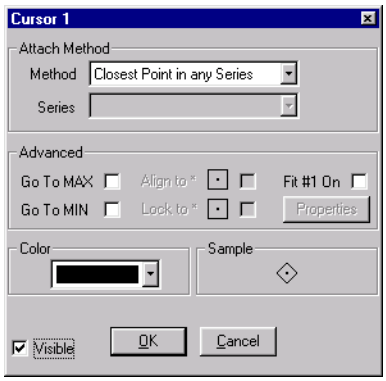
1. In the **Cursor** area of the **Graph** tab Cursors window, select any or all of the cursors by checking the **Visible** checkbox next to each desired cursor (this action simultaneously checks the neighboring **Show Position** checkbox).

NOTE *Unless the **Properties** setting of a cursor is configured otherwise (refer to the next step), you can attach the cursor to any plot line on the graph using the pointing device (mouse). Therefore, you can attach multiple cursors to a single plot.*

---

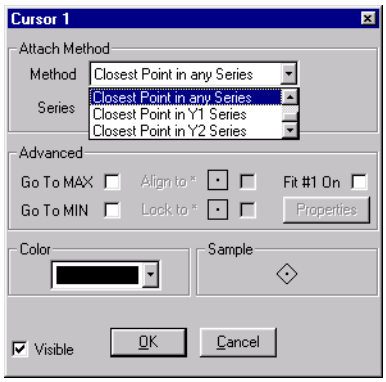
2. Next to the first **Visible** checkbox that you checked, click the **Properties** button. The Cursor <CursorNumber> window opens for the selected cursor.
   Figure 6-273 shows the Cursor 1 window.

Figure 6-273
Example of a Cursor <CursorNumber> window



3. Using the **Cursor <CursorNumber>** window, configure the cursor as follows:
   a. In the **Attach Method** area, select the cursor attachment method in the **Method** combo box. Figure 6-274 shows the options.

Figure 6-274
Cursor attachment method options

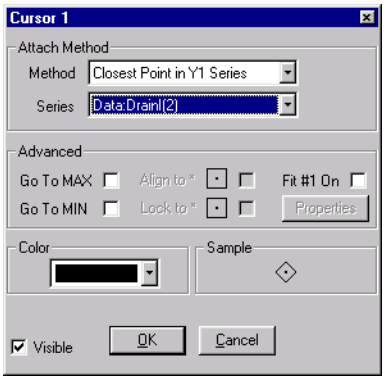


The meanings of these selections are as follows:

- **Closest Point in Any Series:** Allows you to attach the selected cursor to any plot on the graph.
- **Closest Point in Y1 Series**: Only allows you to attach the selected cursor to the specific **Y1** axis plot that is selected in the **Series** combo box (refer to the next substep).
- **Closest Point in Y2 Series:** Only allows you to attach the selected cursor to the specific **Y2** axis plot that is selected in the **Series** combo box (refer to the next substep).

b. If, in step 3a, you chose **Closest Point in Y1 Series** or **Closest Point in Y2 Series**, then proceed to the **Series** combo box and select the specific plot that you want the cursor to attach to. Figure 6-275 shows the **Series** options for **Y1** in the vds-id graph.

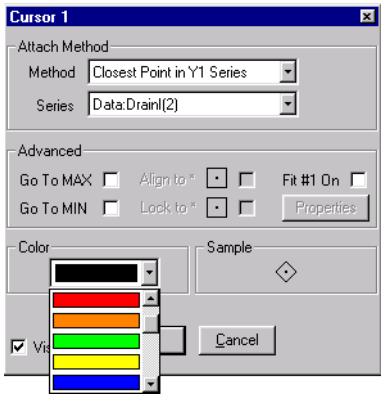
Figure 6-275
Y1 Series plot selections for cursor attachment



c. In the **Color** combo box, select a special color for the cursor (the **Color** selection in the **Cursor <CursorNumber>** window does not affect the color of the cursor-coordinate text block). Figure 6-276 displays some of the available colors.

NOTE *Not all cursor colors can be displayed on all background colors.*

Figure 6-276
Some of the available cursor colors



NOTE *The **Sample** area displays the cursor that you are configuring, including the color.*

d. If you do not want the cursor to display immediately, uncheck the **Visible** checkbox. You can later restore the cursor (the cursor retains its configuration when you uncheck the **Visible** checkbox).

NOTE *The cursor coordinate text block is displayed in the **Display Sample** area. The cursor type number appears next to the coordinate text. However, in the graph, the cursor symbol will appear next to the coordinate text.*

4. Repeat steps 2 and 3 for the other cursors that you selected in step 1.
5. Click **OK**. The cursors and the cursor-coordinate text block now appear on the graph.

NOTE *When you first display the cursors, the default location of the cursors is at the origin, and the default location of the cursor coordinate text block is in the lower right corner of the graph.*

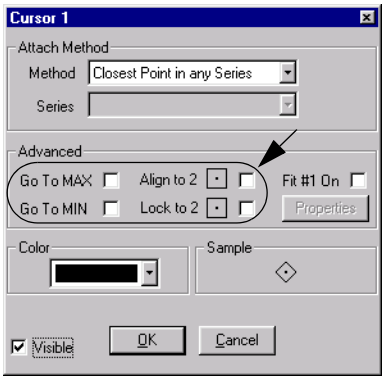
Positioning cursors on the graph using drag-and-drop

6. Position a cursor using **drag-and-drop** as follows:
7. Click on the cursor with the left button of the mouse or other pointing device and continue holding the button down.
8. Drag the cursor to the desired position on a plot.
9. If in step 3a you selected **Closest Point in Any Series** AND the cursor is not on the desired plot, then do the following:
   a. Drag the cursor from the present plot to the desired plot until it attaches to the plot.
   b. On the desired plot, drag the cursor to the desired position.
10. Release the pointing device button. The cursor stays where you left it, and the cursor-coordinate text block displays the coordinates.

Positioning cursors on the graph using special options

The **Advanced** area of a **Cursor** <**CursorNumber**> window provides four checkbox options, which place the <**CursorNumber**> cursor at special locations on the graph. See Figure 6-277.

Figure 6-277
The Advanced cursor-positioning checkbox options

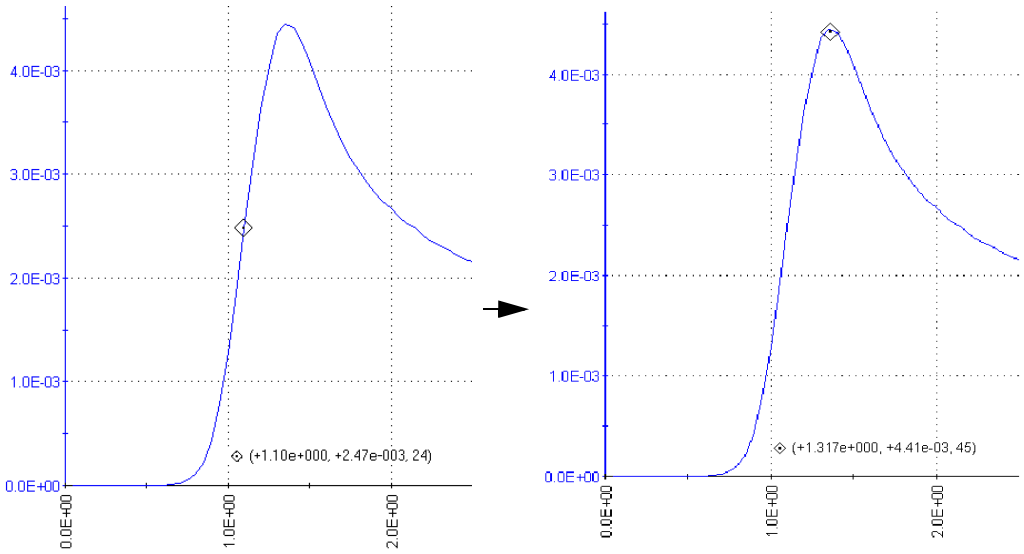


NOTE *The **Align to <CursorNumber>** and **Lock to <CursorNumber>** checkbox options are enabled only when both cursors 1 and 2, both cursors 3 and 4, or both cursors 5 and 6 are active.*

The four checkbox options act as follows:

- **Go To MAX**: Places the **<CursorNumber>** cursor at the maximum-Y data point of the plot to which the cursor is attached. See Figure 6-278.

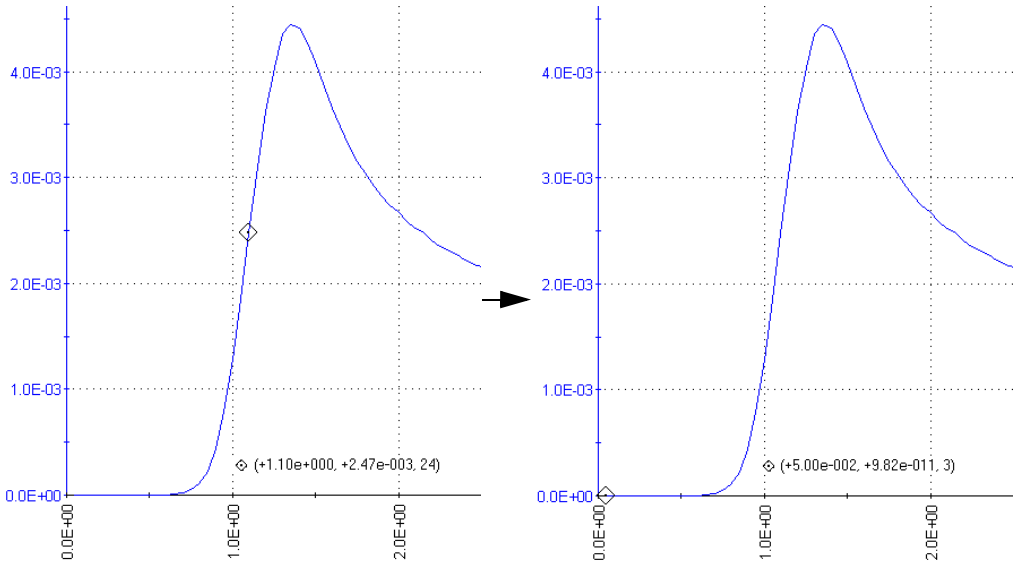
Figure 6-278
Go To MAX operation



After the cursor moves to the maximum point, the **Go To MAX** checkbox reverts to the unchecked state, and the cursor may be manually repositioned.

- **Go To MIN**: Places the **<CursorNumber>** cursor at the minimum-Y data point of the plot to which the cursor is attached. See Figure 6-279.

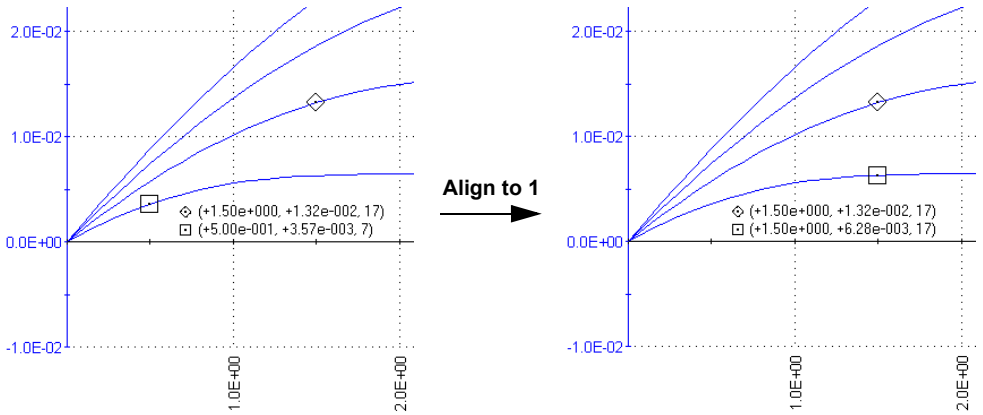
Figure 6-279
Go To MIN operation



After the cursor moves to the minimum point, the **Go To MIN** checkbox reverts to the unchecked state, and the cursor may be manually repositioned.

- **Align To <MatingCursorNumber>**: Aligns the <CursorNumber> cursor to the same X axis value as the <MatingCursorNumber> cursor.[12] See Figure 6-280.

Figure 6-280
Align To <MatingCursorNumber> cursor example

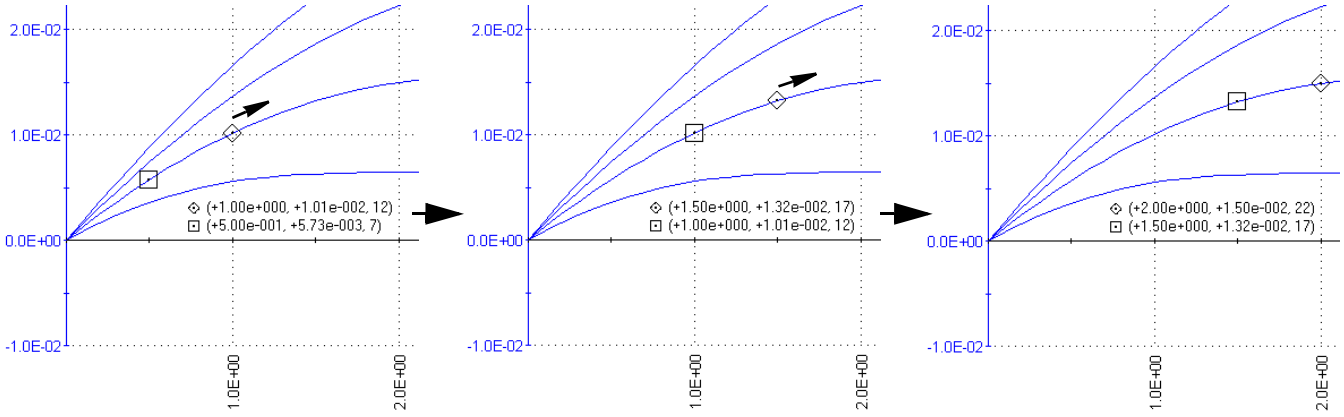


- Nonrepetitive. After the alignment, 1) the **Align To <MatingCursorNumber>** checkbox reverts to the unchecked state, and 2) the cursor may be manually repositioned.

NOTE *The **Align To <MatingCursorNumber>** checkbox is disabled—gray—if the graph does not display the mating cursor when the **Visible** checkbox for the cursor is not checked.*

- **Lock To <MatingCursorNumber>:** Locks the position of the **<CursorNumber>** cursor relative to the position of its mating cursor—the **<MatingCursorNumber>** cursor[12]. The **<CursorNumber>** cursor tracks the movement of the mating cursor, and the relative X distance between the two cursors remains constant. See Figure 6-281.

Figure 6-281
Tracking of diamond cursor by square cursor when its Lock To <MatingCursorNumber> checkbox is checked



12. The **<MatingCursorNumber>** cursor is cursor 2 if **<CursorNumber>** is 1, cursor 1 if **<CursorNumber>** is 2, cursor 4 if **<CursorNumber>** is 3, and so on.

However, the converse is not true; the mating cursor does not track the movement of the <CursorNumber> cursor.

NOTE *The **Lock To <MatingCursorNumber>** checkbox is disabled (gray) if the graph does not display the mating cursor (when the **Visible** checkbox for the cursor is not checked).*

Performing line fits between cursors

The **Fit #1 (**alternatively **Fit #2)** checkbox and the **Properties** button of a Cursor **<CursorNumber>** window may be used to initiate a line fit between existing cursors. For information on using these functions, refer to Performing line fits using existing cursors.

Formatting the displayed coordinates

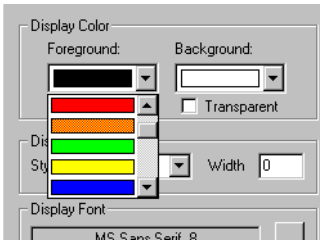
The next four subsections describe how to change the colors, border, and font of the cursor coordinate text block.

Changing cursor coordinate text color

You can change the color of the coordinate text as follows:

1. Under **Display Color** in the **Graph** tab Cursors window, click the scroll arrow on the **Foreground** combo box. Several color selections are available, some of which are shown in Figure 6-282.

Figure 6-282
Some of the available coordinate text colors



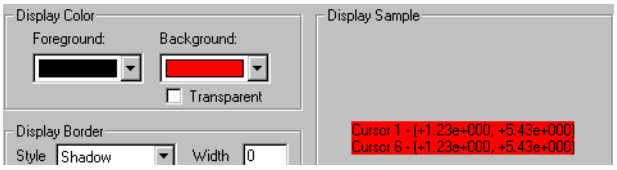
2. Select the desired text color. The **Display Sample** area displays the coordinate text in the selected color.
3. Click **OK**. The new coordinate-text color is displayed in the graph.

Changing background color of the cursor coordinate text block

You can change the background color of the cursor coordinate text block as follows:

1. Under **Background Color** in the **Graph** tab Cursors window, click the scroll arrow on the **Background** combo box. The same color selections as for text color are available (for example, see Figure 6-282).
2. Select the desired text background color. The **Display Sample** area displays the background color. See Figure 6-283.

Figure 6-283
Example of special coordinate text background color



3. Click **OK**. The new coordinate text background color is displayed in the graph.

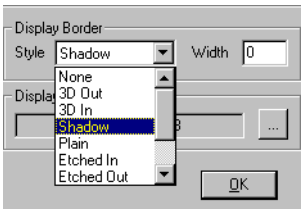
NOTE *If you want other graph components to be able to shine through the normally opaque background of the cursor coordinate text block, select the **Transparent** checkbox on the **Display Color** area of the Cursors window. However, be aware that a checked **Transparent** checkbox 1) overrides the background color selection (signified by a gray color in the **Background** combo box), and 2) causes a border (discussed next) to be displayed in gray scale.*

Adding/changing a border around the cursor coordinate text block

You can add or change a border around the cursor coordinate text block, as follows:

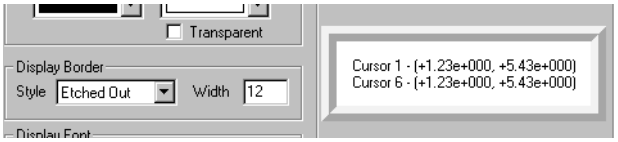
1. Under **Display Border** in the **Graph** tab Cursors window, click the scroll arrow on the **Style** combo box. Several border style selections are available, some of which are shown in Figure 6-284.

Figure 6-284
Some of the available borders for a cursor coordinate text block



2. Select the desired border type (if the adjacent **Width** setting is 0, which is the default, the **Display Sample** area does not yet display a border).
3. Also under **Display Border** in the Cursors window, in the **Width** text box type a width for the border. KITE accepts values between 0 and 20 pixels. The **Display Sample** area displays the selected border around the cursor coordinate text block (in the same color as selected in the **Background** combo box). For example, see Figure 6-285.

Figure 6-285
Example 12-pixel Etched Out border for cursor-coordinate text block



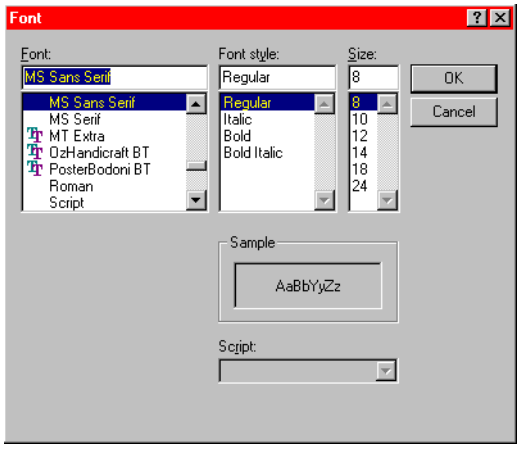
4. Click **OK**. The graph displays the cursor coordinate text block with the selected frame.

Changing the font in a cursor-coordinate text block

Change the cursor coordinate font as follows:

1. Under **Display Font** in the **Graph** tab Cursors window, click the arrow of the combo box. The Font window appears. See Figure 6-286.

Figure 6-286
Font window



2. In the Font window, select the name, size, and style of the font desired for the cursor coordinate text. The **Sample** area of the Font window displays the new font.
3. Click **OK**. The **Display Sample** area of the **Cursors** window displays the cursor coordinate text with the selected font.
4. Click **OK**. The graph displays the cursor coordinate text with the selected font.

Positioning the displayed cursor coordinates

The cursor coordinate text block can be positioned anywhere in the graph, as follows:

1. Click on the text block with the left button of the mouse or other pointing device and continue holding the button down.
2. Drag the text block to the desired position on the graph.
3. Release the pointing device button.

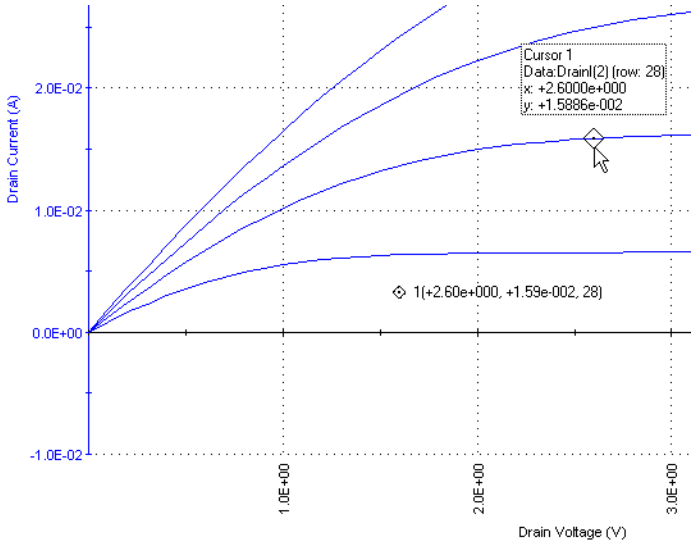
Using the pointing device (mouse) to view information about the cursor-specified data

When you select a cursor with the mouse or other pointing device, KITE displays the following information directly next to the cursor:

- The cursor number.
- The data series.
- The **Data** worksheet row number.
- The cursor coordinates.

This feature 1) facilitates simultaneous viewing of the cursor and its coordinates; and 2) allows you to temporarily view the coordinates at a higher precision than typically chosen for permanent display. See Figure 6-287.

Figure 6-287
Viewing pointing-tool-selected cursor coordinate



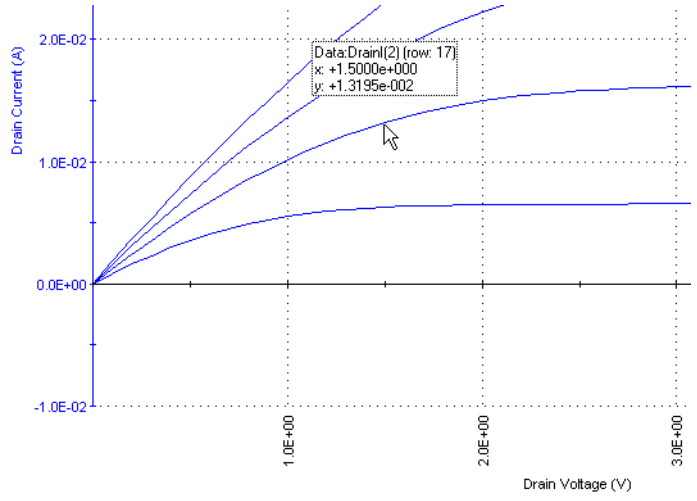
Viewing plot coordinates and data series properties using the pointing device (mouse)

When you select a data point on any plot using the mouse or other pointing device (using a left click), KITE displays the following information about the point:

- Its data series.
- Its **Data** worksheet row number.
- Its coordinates, to four decimal places.

See Figure 6-288.

Figure 6-288
Viewing pointing-tool-selected data coordinates



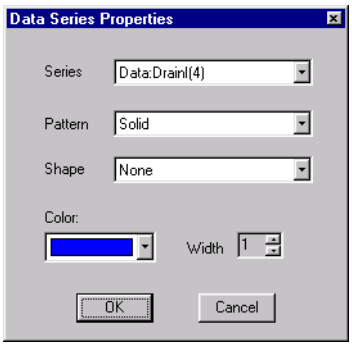
This feature allows you to quickly check information about any point on the graph without using cursors. To display the information, do the following:

1. Place the default graph cursor (⌖) over the plot line at approximately the location of the desired data point.
2. Move the ⌖ cursor along the plot line until it is over the data point, where the ⌖ cursor changes to the *pointer* cursor and the coordinates, and so on display above it.

You can also display additional information about the data series used for the plot. Do the following:

1. After displaying data-point information as follows (approximately the same procedure as above), maintain the pointing-device (mouse) position for step 2.
   a. Place the default graph cursor (⌖) over the plot line.
   b. Move the ⌖ cursor along the plot line until it is over any data point, as indicated by a change of the ⌖ cursor to a pointer cursor and a local display of point coordinates, and so on.
2. Right-click the pointing device (mouse). A Data Series Properties window appears for the plot line. See Figure 6-289.

Figure 6-289
Data Series Properties window

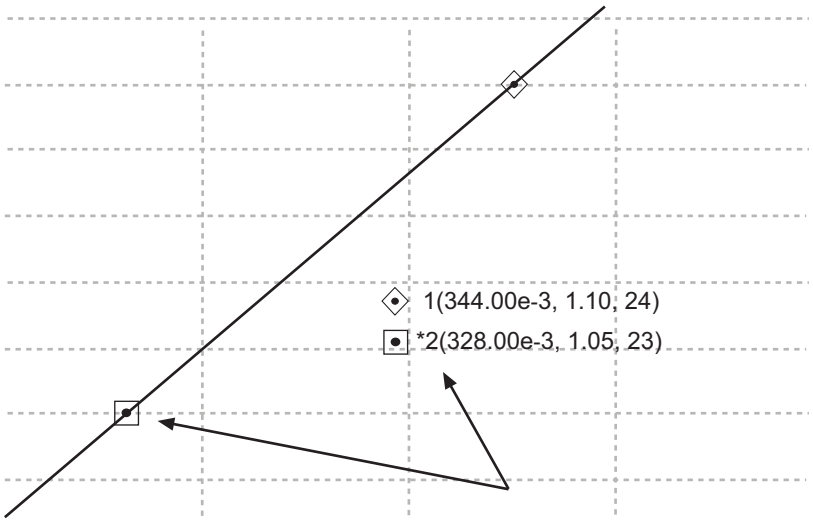


Interpolate data on the graph

A cursor must be selected before the move or interpolation key sequences become active for that cursor. To interpolate data on the graph perform the following procedures:

- Select a cursor, hold down the **Alt** key, and then use the arrow keys to find the desired point you want to interpolate data. When you hold down the **Ctrl** key and use the arrow keys you will only see the data points taken by the SMUs.
  - Cursors can also be selected using the **Tab** key. Pressing the Tab key selects the next cursor, if more than one cursor is displayed.

Figure 6-290
Interpolate cursor functions



Cusors selected by left-clicking
(Note Bolded cursor and * in label)

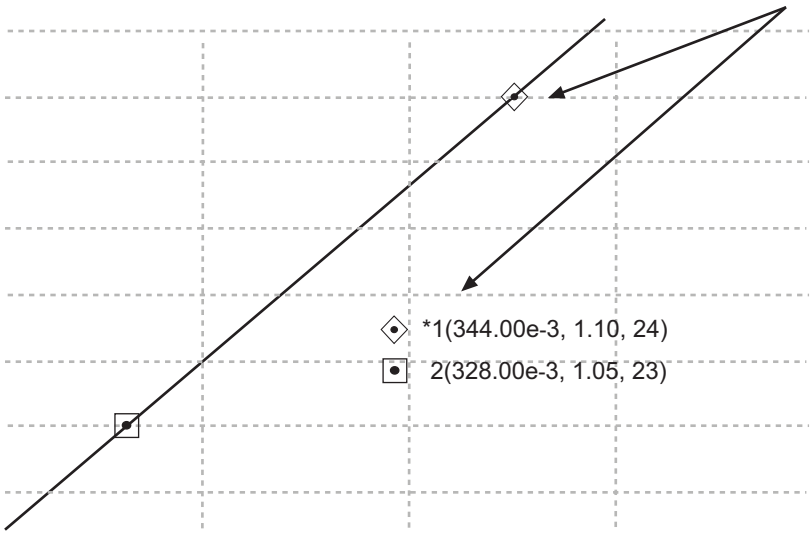
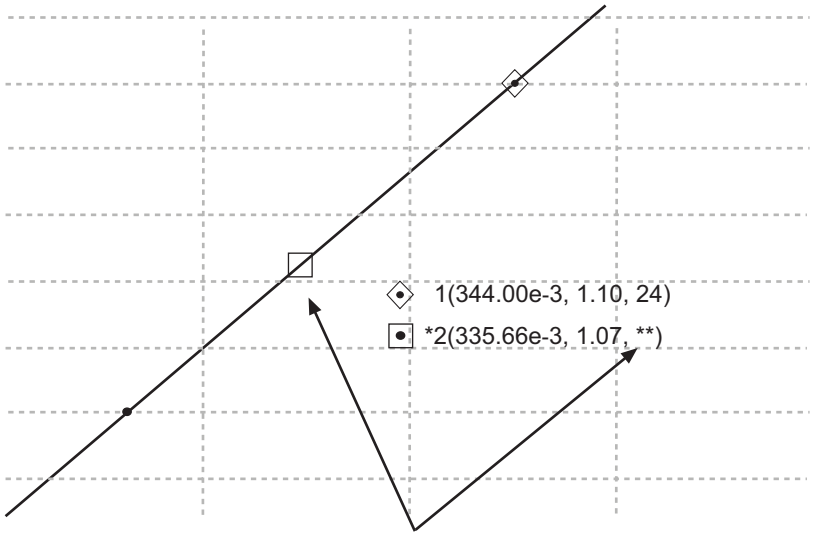


Figure 6-291
Interpolate cursor functions between points



- Holding down the Alt key and using the Right and Left Arrow keys, the user can interpolate between points on the series. The cursor moves 1 pixel at a time in normal mode and 5 pixels at a time in fast mode.

NOTE *Note the high-lighted cursor is between points and the label has a ** where the row number used to be indicating it is an interpolated value.*

- By holding down the Ctrl key and using the Left and Right Arrow keys, the user can move the cursor between points in the series, just like dragging the cursor with the mouse.
- Holding down the Ctrl or Alt and Arrow key for longer than about a second activates a high speed move mode; the cursor will move at a faster rate.

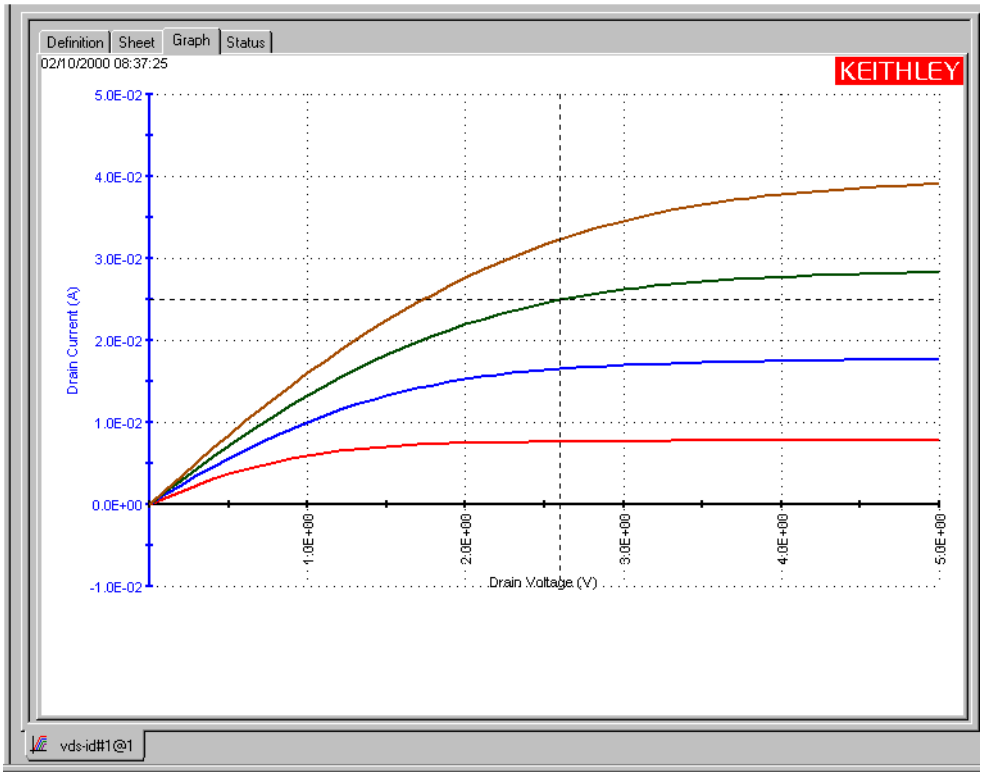
Zoom affect operation

If you zoom in so far that there are no more points displayed, the cursor will no longer move in interpolated or point to point mode.

Visually reading plot coordinates using cross hairs

If you only want a visual aid for determining plot coordinates, you can display a set of cross hairs that can be positioned anywhere on the graph. See Figure 6-292.

Figure 6-292
Crosshair example



Open the crosshairs as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Crosshair**. The crosshairs appear on the graph.

Close the cross hairs in the same way you opened them (in the **Graph Settings** menu, the **Crosshair** selection toggles).

NOTE *You can use to quantify zoomed parts of a graph. Refer to Temporarily enlarging a selected area of the graph by zooming.*

Performing on-graph line fits

The **Line Fits** item in the **Graph Settings** menu allows you to directly fit lines to **Graph** tab plots. Up to two fits may be performed on the graph, selected from among the following types:

- **Linear**: Chord line of the form $y = a + bx$, drawn between two graphically defined data points.
- **Regression**: Regression line of the form $y = a + bx$ for a graphically defined range of data points.
- **Exponential**: Regression line of the form $y = a \cdot e^{bx}$ for a graphically defined range of data points.
- **Log**: Regression line of the form $y = a + b \cdot log_{10}(x)$ for a graphically defined range of data points.
- **Tangent**: Tangent to the plot at a graphically defined data point. The tangent line has the form $y = a + bx$.

The **Graph** tab displays the following:

1. the fitted line.
2. the fit parameters.
3. the tangent data point[13] or the starting and ending data points (data range).
4. the data-point coordinates. Tangent or starting and ending data points are defined by cursors.[14]

The results obtained with **Graph** tab line fits are similar to the results obtained with the corresponding **Formulator** functions, as shown in Table 6-12 below.

Table 6-12
Correspondence between Graph tab and Formulator line fits

Graph tab fit	Formulator fits that return the corresponding fit line and fit parameters			
	Fit line	Fit parameter "a"	Fit parameter "b"	Fit parameter "xint"
Linear	LINFIT	LINFITYINT	LINFITSLP	LINFITXINT
Regression	REGFIT	REGFITYINT	REGFITSLP	REGFITXINT
Exponential	EXPFIT	EXPFITA	EXPFITB	Not applicable
Log	LOGFIT	LOGFITA	LOGFITB	Not applicable
Tangent	TANFIT	TANFITYINT	TANFITSLP	TANFITXINT

However, the **Graph** tab and **Formulator** tools each provide specific advantages. For example, **Graph** tab fits facilitate visual what if trials on various data point(s), whereas **Formulator** fit results can be used directly in other calculations.

Line fit examples

Figures Figure 6-293 through Figure 6-297 illustrate the five line fit types.

13. The data point at which a **Tangent** line is fitted to the plot.
14. For more information about cursors, refer to Numerically displaying plot coordinates using cursors.

Figure 6-293
Linear fit example

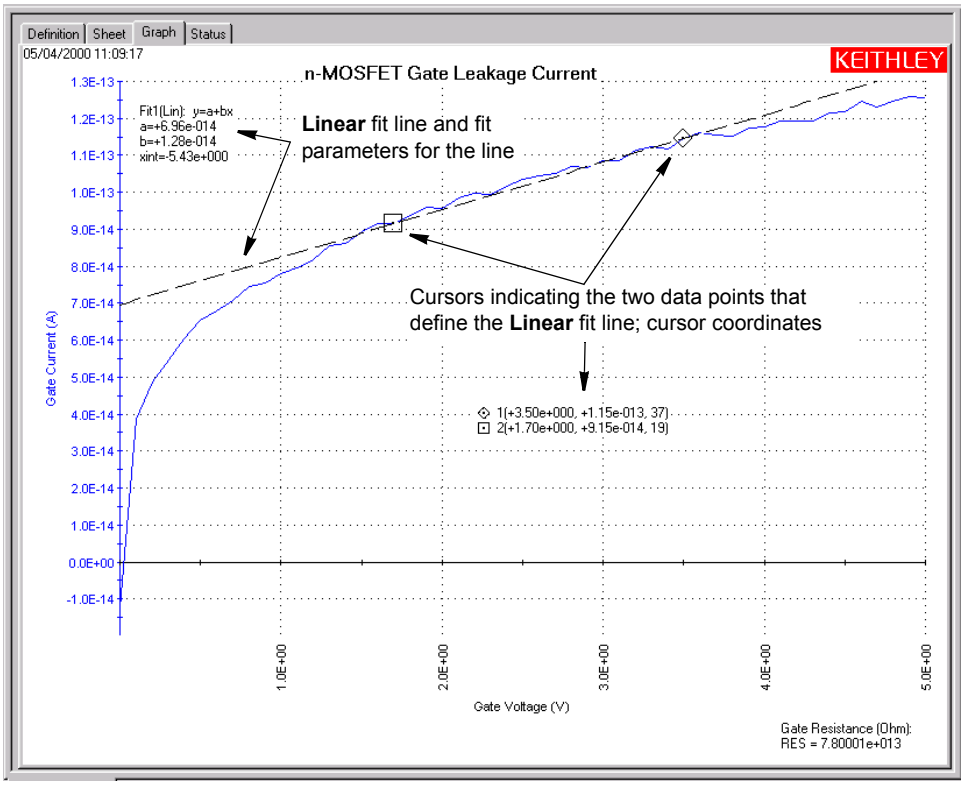


Figure 6-294
Regression fit example

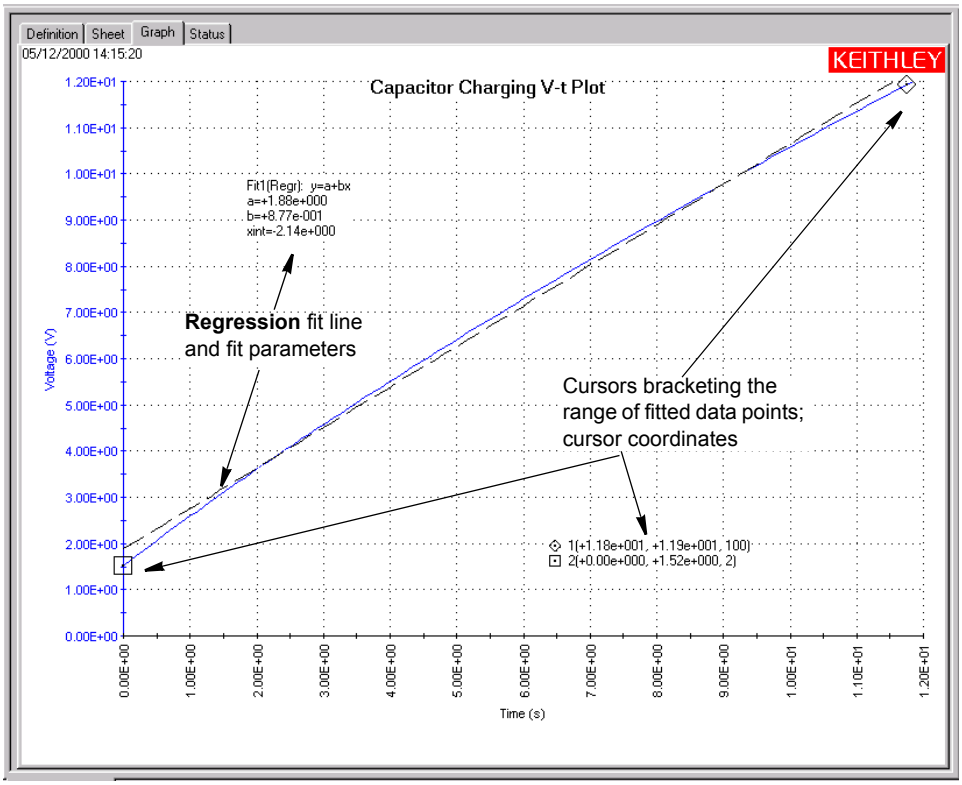


Figure 6-295
Exponential fit example

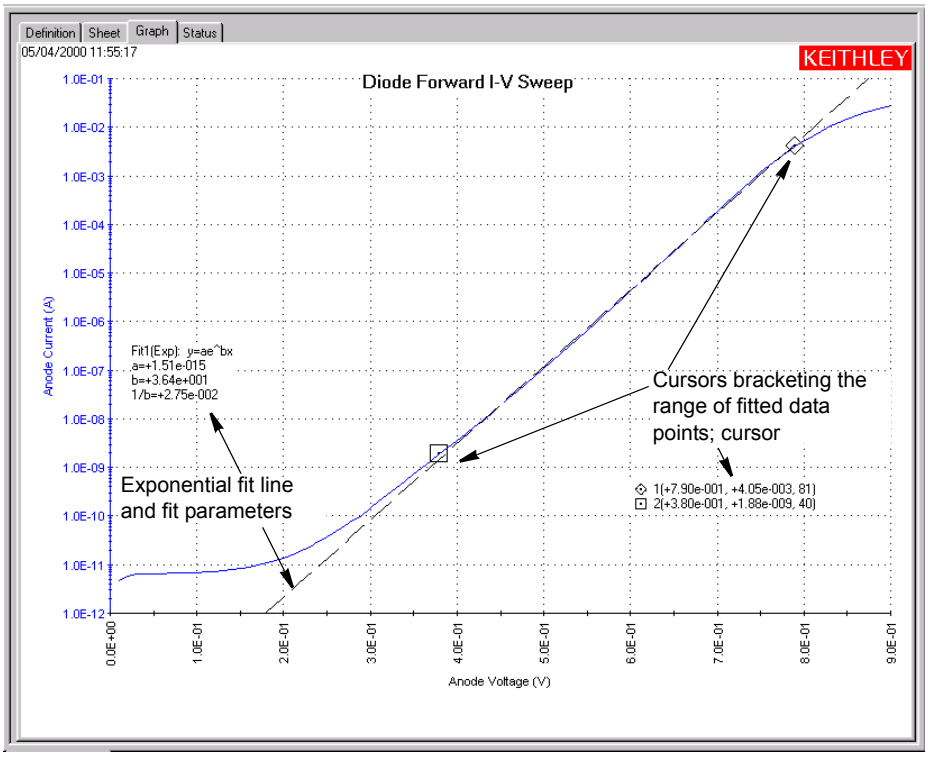


Figure 6-296
Log fit example

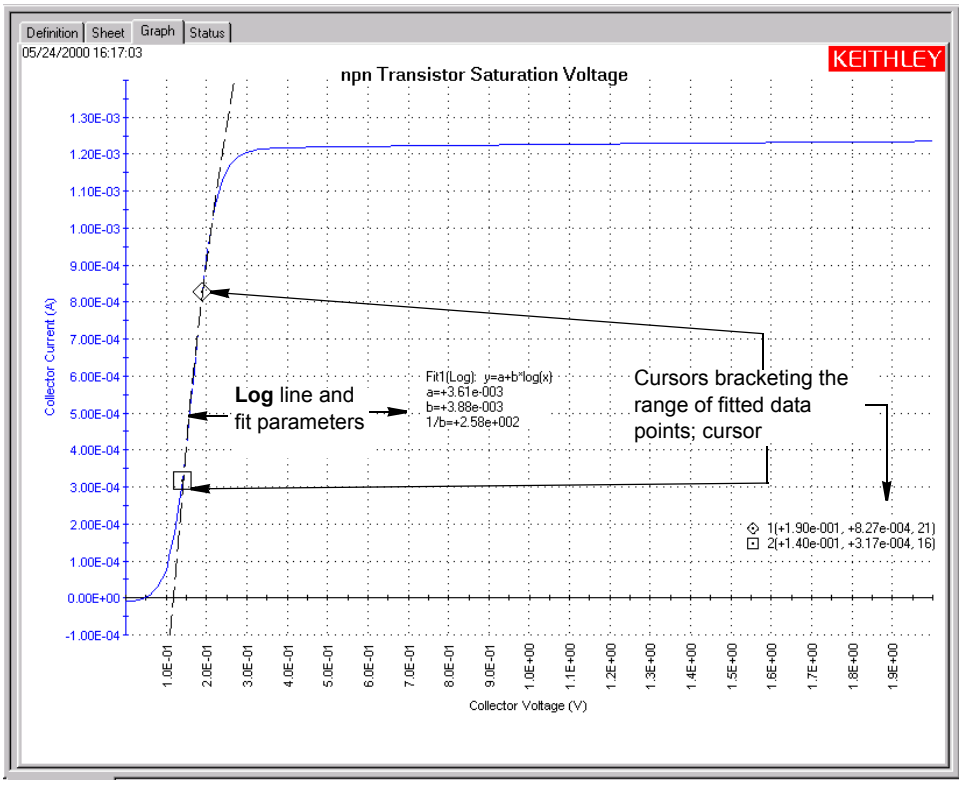
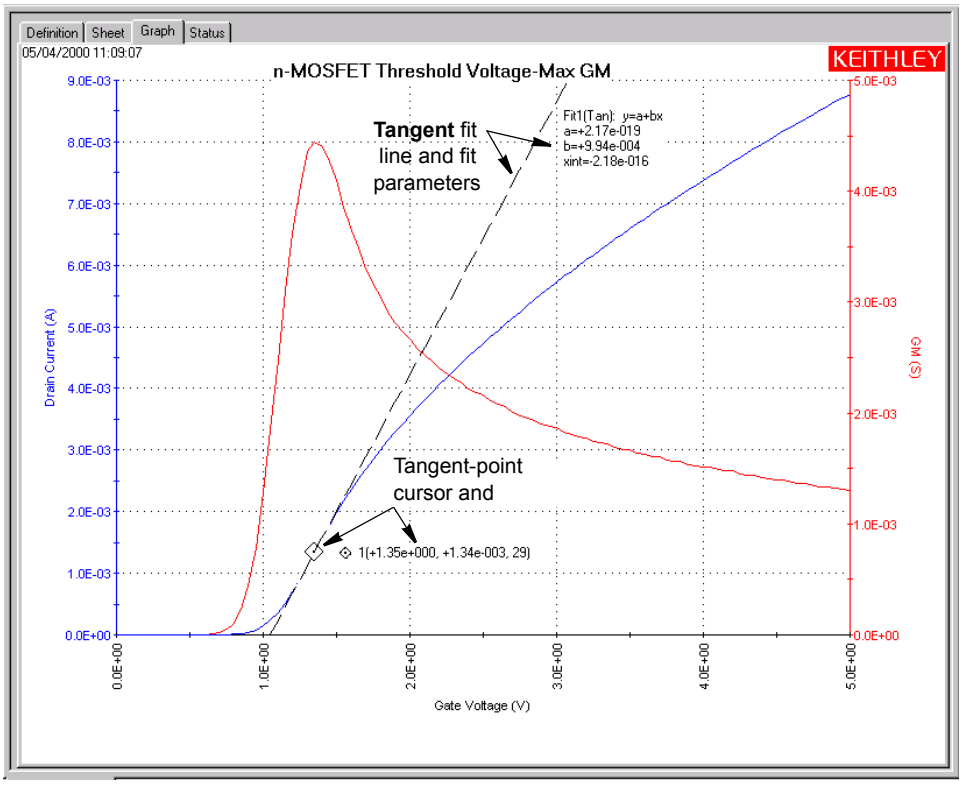


Figure 6-297
Tangent fit example

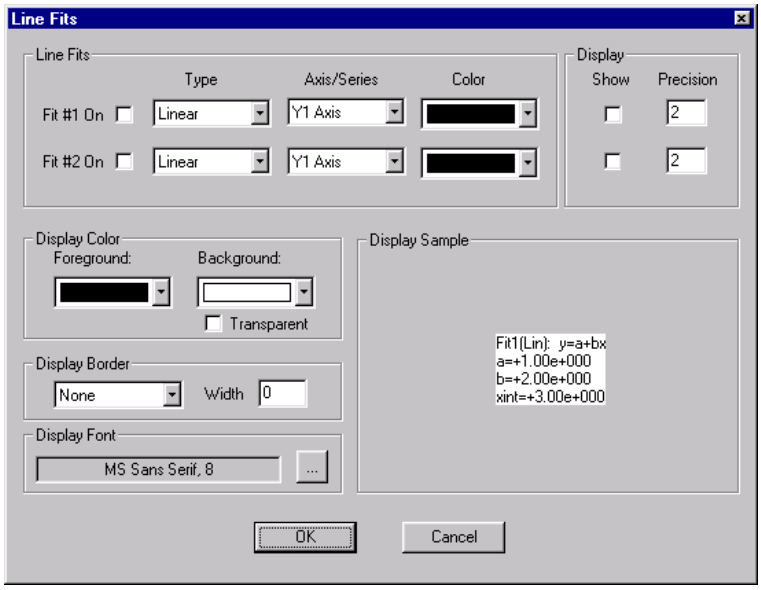


Performing fits

Perform fits as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Line Fits.** The Line Fits window opens. See Figure 6-298.

Figure 6-298
Line Fits window



3. In the Line Fit Properties window, select the fit as follows:
   a. Enable the fit by checking the **Fit #1 On** or **Fit #2 On** check box, as appropriate.

NOTE *Fit #1 is always associated with cursors 1 and 2. Fit #2 is always associated with cursors 3 and 4.*

---

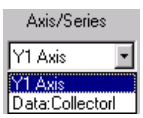
   b. In the corresponding **Type** combo box, select the type of fit. See Figure 6-299.

Figure 6-299
Type combo box selections

   c. In the corresponding **Axis/Series** combo box, select the data series for which the fit is to be made; or, alternatively, the Y axis to which you want to reference free-floating points (refer to the subsequent bulleted list). Figure 6-300 shows examples for a graph with one Y-parameter and one Y axis. For a graph with more parameters, the menu displays selections for each parameter. For a graph with two Y axes, the menu also displays a Y2 axis selection for **Linear** fits only.

Figure 6-300
Examples of Axis/Series combo box selections

Linear fit axis selections



Regression, **Exponential**, **Log**, and **Tangent** fit axis selections

The following applies to selecting a data series or axis:

- Selecting a data series results in display of two cursors that attach to the specified data curve.You may select a data series for any type of fit.
- Selecting a Y axis results in the display of free-floating fit cursors that may be positioned anywhere on the graph. Fit parameters reflect the scale of the selected Y axis. You may select a Y axis (for example, **Y1 Axis** in Figure 6-300) only for a **Linear** fit.

d. In the corresponding **Color** combo box, select the color of the fit line.

NOTE *Unique, extra-long dashes distinguish a fit line from all types of plot lines. Therefore, a special color is not normally essential.*

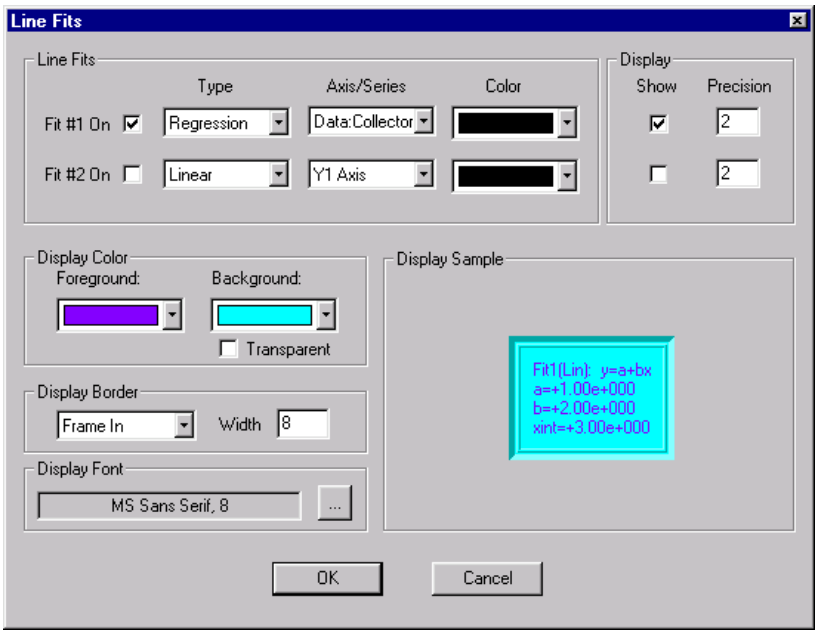
---

e. In the **Display** area, check the **Show** checkbox to enable display of the fit parameters,

f. To display fit parameters with greater or lesser precision than two decimal places, change the **Precision** value

4. To add a second fit, repeat the substeps under step 3 for **Fit 2 (**or **Fit 1,** if **Fit 2** was created first).
5. If desired, change the color, border, and font of the fit parameter display(s), referring to Formatting the displayed fit parameters. Figure 6-301 illustrates examples of each setting in the Line Fits window.

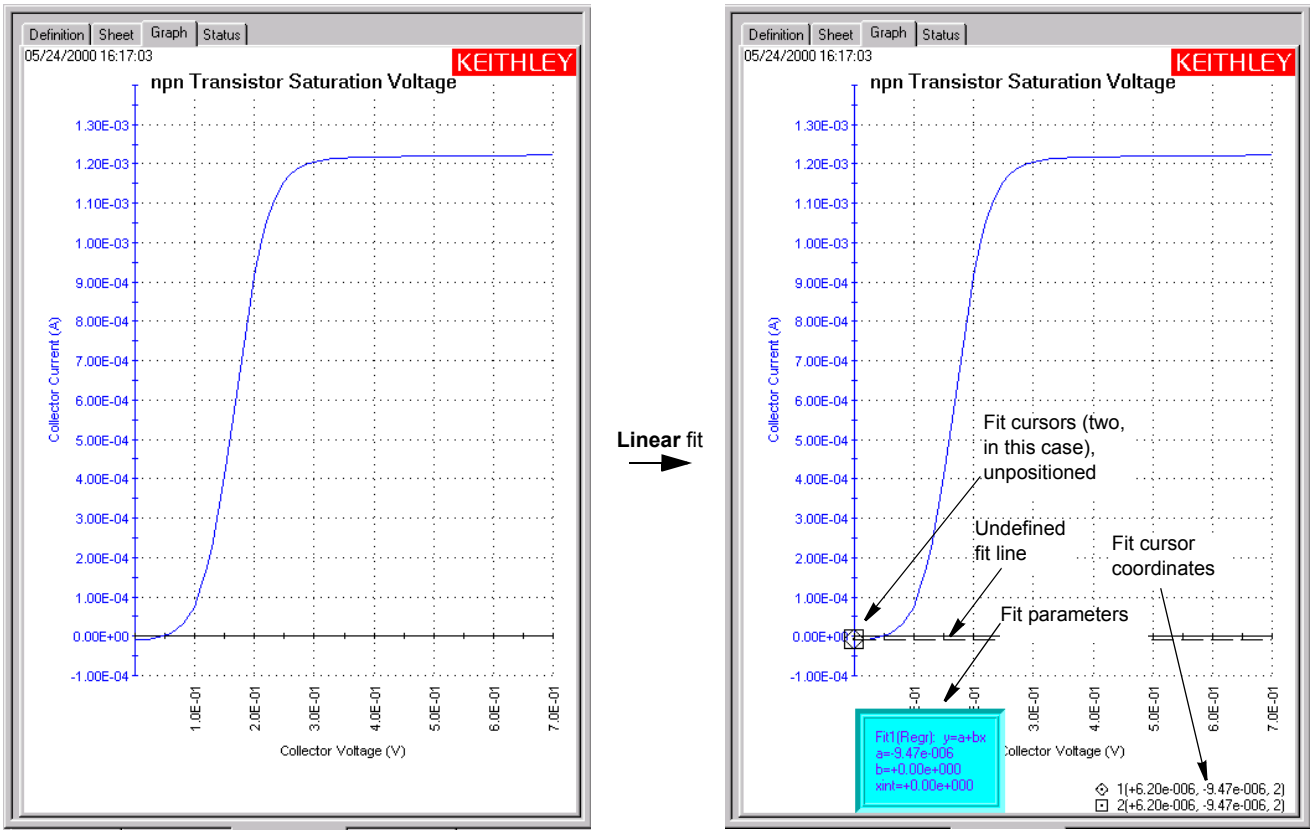
Figure 6-301
Line Fits window setting examples



6. Click **OK**. The line-fit routine executes, the Line Fits window closes, and the graph displays the following new items for *each* fit:
   - Two new cursors (one for a **Tangent** fit) are temporarily located at the origin or on a Y axis or both. In the next step, you will position these cursors to define the starting and ending data points (data range) for the fit or the data point at which a tangent is to be fit, as
     applicable.
   - Numeric fit parameter display.
   - Numeric cursor coordinate display.

Figure 6-302 illustrates the result of the settings shown in Figure 6-301.

Figure 6-302
Example of initial fit result



NOTE *The dashed-line plot of the fit line appears only after completing the next step.*

7. Adjust the cursor locations as follows, using the methods described under Positioning cursors on the graph using drag-and-drop or Positioning cursors on the graph using special options.

NOTE *Positively specify each cursor location. If the step 6 temporary location for a cursor (for example, the origin) is also the desired location, inform KITE by moving the cursor away from that location and then back again.*

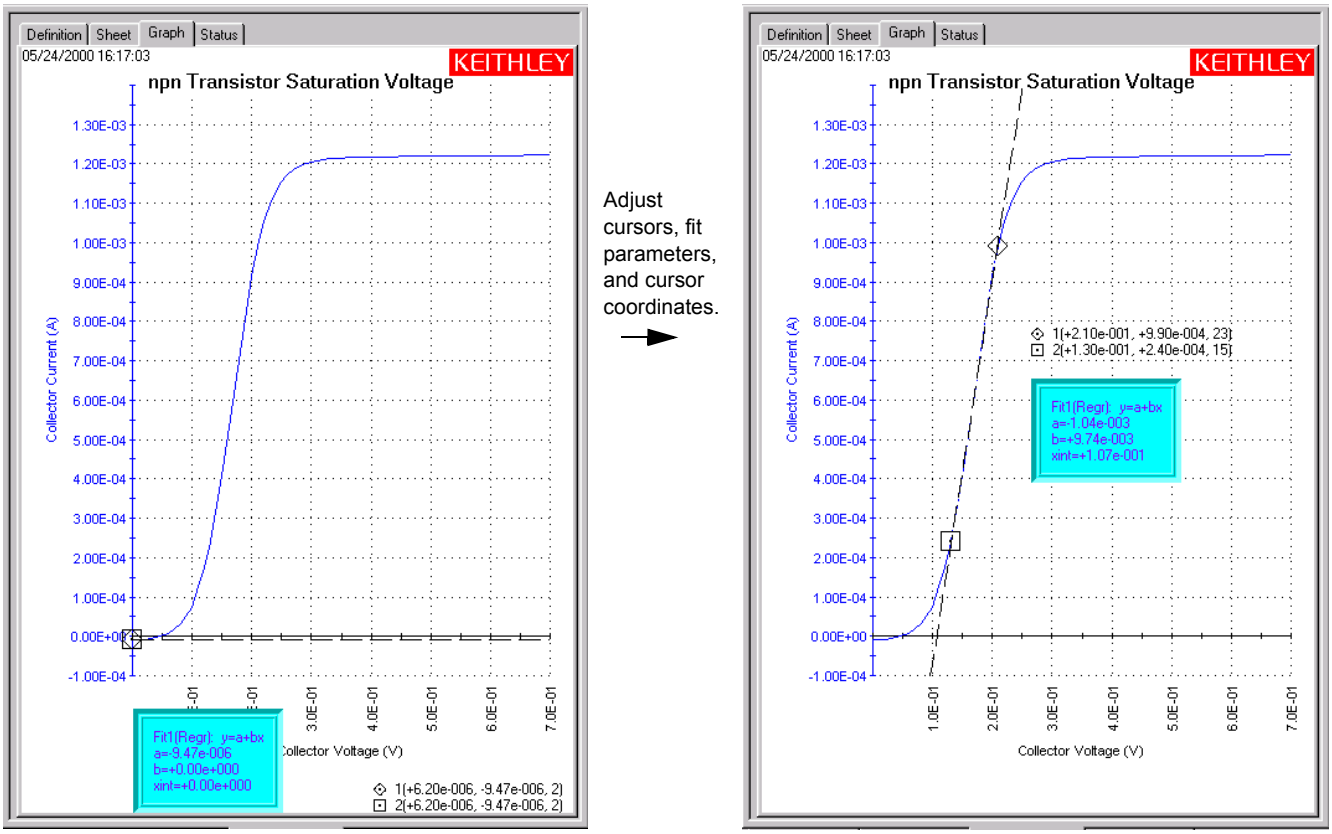
- For a **Linear** fit, adjust the two cursors that define the two points through which the fit line is to be drawn.
- For a **Regression**, **Exponential**, or **Log** fit, adjust the two cursors that define the range (the starting and ending points) of the data to be fitted.
- For a **Tangent** fit, place the cursor on the point against which you want the tangent to be fitted.

Plots of the fit lines appear as dashed lines, and fit parameter and cursor coordinate displays indicate appropriate numerical values. Figures 6-293 through 6-297 illustrate typical fits (following relocation of the fit parameter and cursor coordinate displays, as described subsequently in Changing the position of the displayed fit parameters).

8. Reposition the displays of fit parameters and cursor coordinates, as required.

Figure 6-303 illustrates a fit after positioning the fit cursors, the fit parameters, and the fit cursor coordinates.

Figure 6-303
Example of finished fit result



Formatting the displayed fit parameters

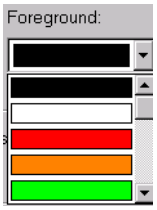
This subsection provides details about specifying/changing the colors, frame, and font, and position of the displayed fit parameters.

Changing the text color of the displayed fit parameters

Change the color of the fit parameter text as follows:

1. Under **Display Color** in the Line Fit Properties window, click the scroll arrow on the **Foreground** combo box. Several color selections are available, some of which are shown in Figure 6-304.

Figure 6-304
Some of the available fit-parameter text colors



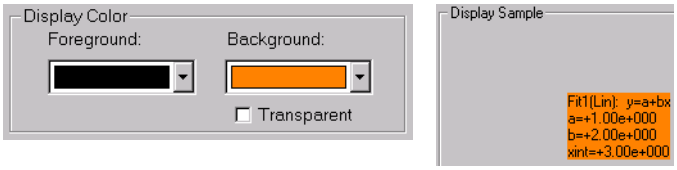
2. Select the desired text color. The **Display Sample** area displays the fit parameter text in the selected color.
3. Click **OK**. The new fit parameter text color is displayed on the graph.

Changing background color of the displayed fit parameters

Change the background color for the displayed fit parameters as follows:

1. Under **Color** in the **Graph** tab Line Fit Properties window, click the scroll arrow on the **Background** combo box. Several color selections are available; they are the same colors for text color. For examples, see Figure 6-304.
2. Select the desired text background color. The **Display Sample** area displays the background color. See Figure 6-305.

Figure 6-305
Example of fit parameters background color



3. Click OK. The new data variable background color is displayed in the graph.

NOTE *If you want other graph components to be able to shine through the normally opaque background of the displayed fit parameters, select the **Transparent** checkbox on the **Color** area of the Line Fit Properties window. However, be aware that a checked **Transparent** checkbox:*

- *overrides the background color selection (signified by a gray color in the **Background** combo box).*
- *causes a border to be displayed in gray scale.*

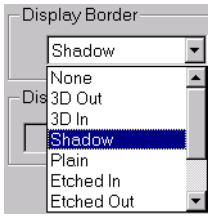
For more about borders, refer to the following paragraph.

Adding/changing a border around the displayed fit parameters

You can add or change a border around the displayed fit parameters as follows:

1. Under **Border** in the **Graph** tab Line Fit Properties window, click the scroll arrow on the **Display Border** combo box. Several border style selections are available, some of which are shown in Figure 6-306.

Figure 6-306
Some of the available borders for displayed fit parameters

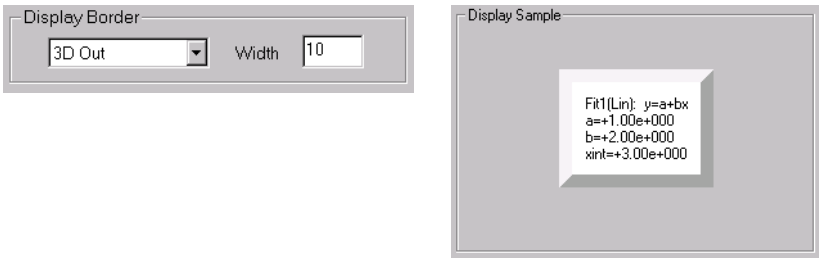


2. Select the desired border style.

NOTE *If the adjacent **Width** setting is "**0**" (the default), the **Sample** area does not yet display a border.*

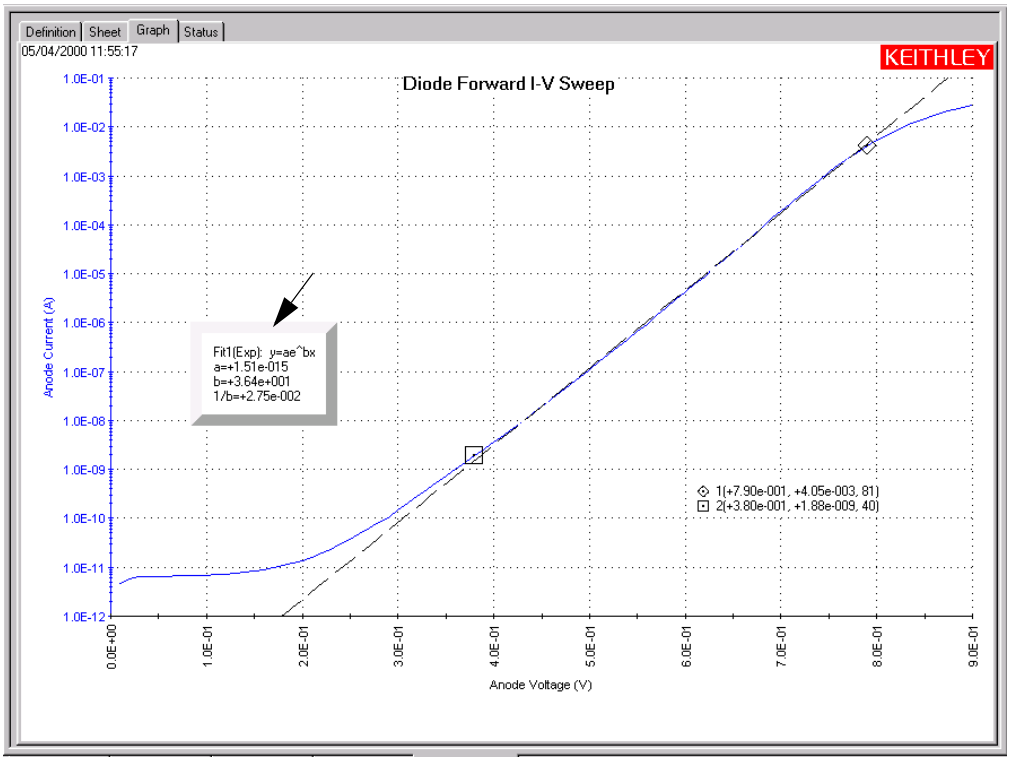
3. In the adjacent **Width** text box, type a width for the border. KITE accepts values between 0 and 20 pixels. The **Display Sample** area displays the selected border around the values, in the same color as selected in the **Background** combo box. See Figure 6-307.

Figure 6-307
Example 10-pixel 3D Out border for displayed fit parameters



4. Click OK. The graph displays the fit parameters with the selected border. Figure 6-308 shows a shadow-bordered display of **"vfd"** fit parameters.

Figure 6-308
Display of 3D Out-bordered "vfd" fit parameters

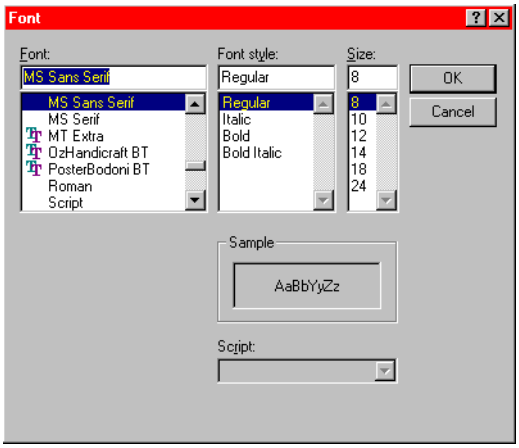


Changing the font of the displayed fit parameters

Change the fit parameters font as follows:

1. In the **Graph** tab Line Fit Properties window, click the button at the right side of the **Font** combo box. The Font window appears. See Figure 6-309.

Figure 6-309
Font window



2. In the Font window, select the name, size, and style of the font desired for the fit parameters text. The **Sample** area of the Font window displays the new font.
3. Click **OK**. The **Sample** area of the line Fit Properties window displays the fit parameters with the selected font.
4. In the Line Fit Properties window, click **OK**. The graph displays the fit parameters with the selected font.

Changing the position of the displayed fit parameters

The displayed fit parameters and cursor coordinates can be positioned anywhere on the graph, as follows:

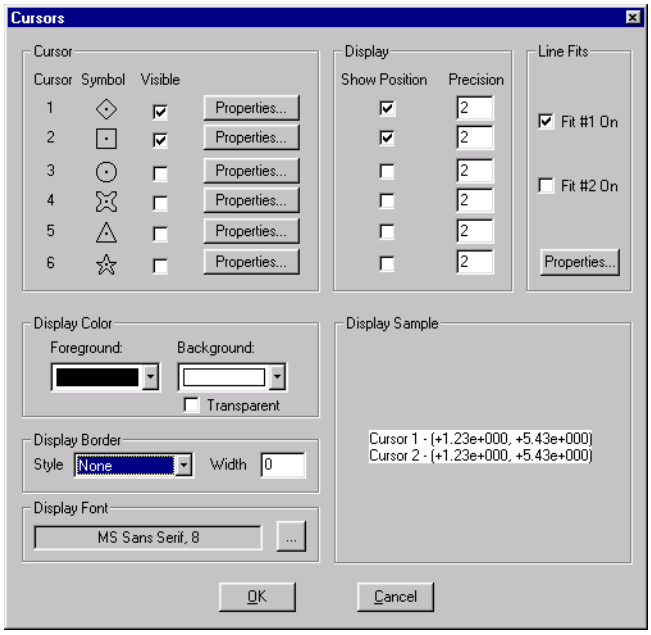
1. On the graph, click on a block of displayed fit parameters or cursor coordinates with the left button of the mouse or other pointing device; continue holding the button down.
2. Drag the displayed parameters/coordinates to the desired position on the graph.
3. Release the pointing device button.

Formatting and positioning the fit cursor coordinates

Format the fit cursor coordinates as follows:

1. Right click the cursor coordinates display. The Cursors window opens. See Figure 6-310.

Figure 6-310
Cursors window example



2. Change cursor coordinate format in essentially the same way as for fit parameters. For details, refer to Formatting the displayed coordinates.

Change the cursor coordinate position by drag-and-drop, just as for fit parameters (refer to Changing the position of the displayed fit parameters.

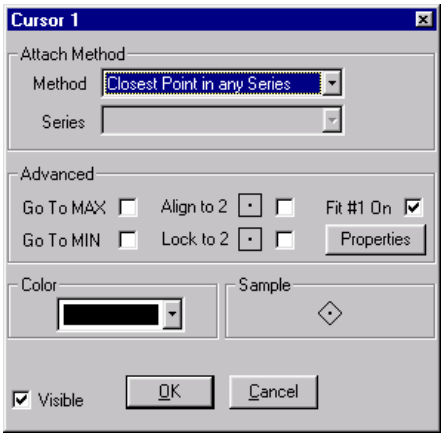
Performing line fits using existing cursors

NOTE *Only cursors 1 through 4 may be used for line fits.*

To use existing cursors for fits, do the following:

1. Right-click a cursor that is to be used for a line fit. The Cursor **<CursorNumber>** window for that cursor opens. See Figure 6-311.

Figure 6-311
Cursor <CursorNumber> window example



2. In the Cursor **<CursorNumber>** window, check the **Fit #1** or **Fit #2** checkbox, whichever is displayed.

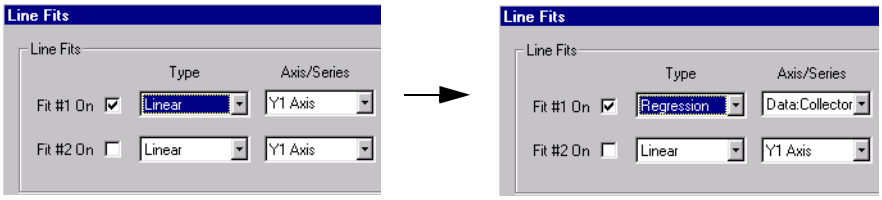
NOTE *Fit #1 is always associated with cursors 1 and 2. Fit #2 is always associated with cursors 3 and 4.*

You can alternatively use the Cursors window to specify that an existing cursor(s) is to be a line fit cursor(s). Open the Cursors window (Figure 6-310 above) using the following sequence: ***Tools > Graph Settings*** *menu* ***> Cursors.*** *Then check* ***#1 On*** *or* ***#2 On****, as appropriate.*

---

3. In the Cursor **<CursorNumber>** window, click the **Properties** button. The Line Fits window opens.
4. In the Line Fits window, under **Type,** select the desired line fit type.
   For illustration purposes, Figure 6-312 shows **Regression** being selected in place of the default (**Linear**).

Figure 6-312
Regression line-fit selection

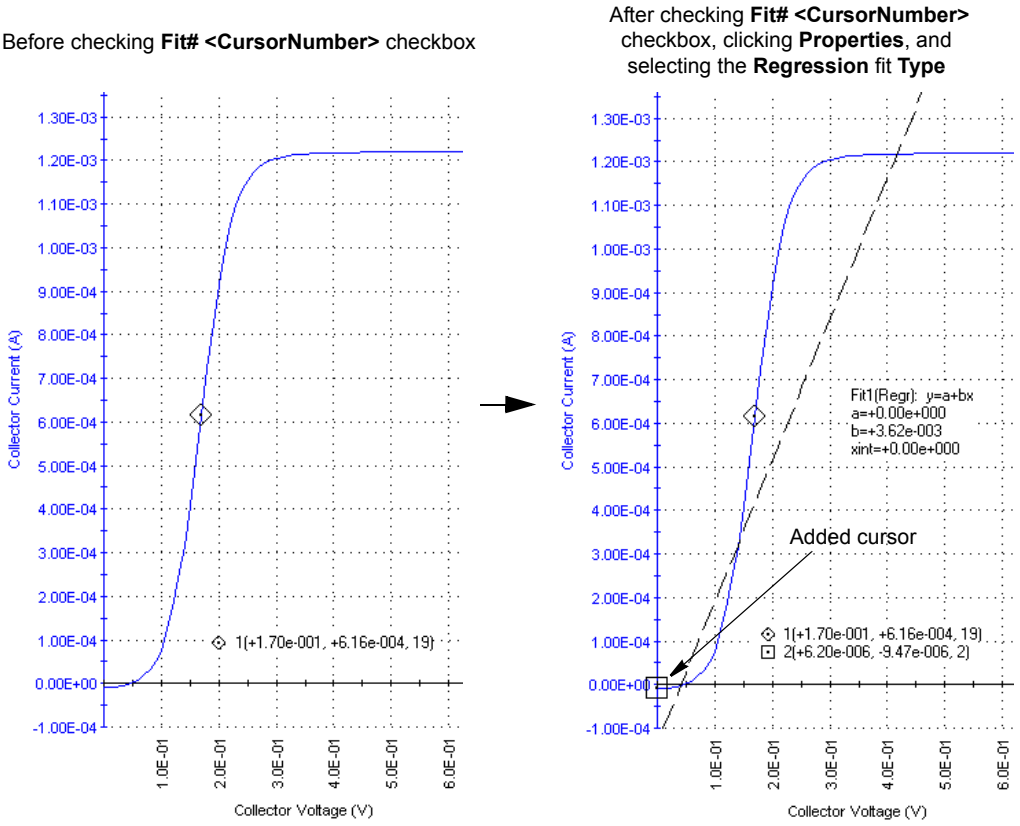


5. In the Line Fits window, click **OK**.

6. In the still open Cursor **<CursorNumber>** window, click **OK**.
   The following new items appear on the graph:
   - An additional mating cursor, if a mating cursor was not already displayed when you started this procedure (cursor 2 if **<CursorNumber>** is 1; cursor1 if **<CursorNumber>** is 2, cursor 4 if **<CursorNumber>** is 3, and cursor 3 if **<CursorNumber>** is 4).
   - Coordinates for both cursors.
   - A fit line.
   - Fit parameters corresponding to the line fit.

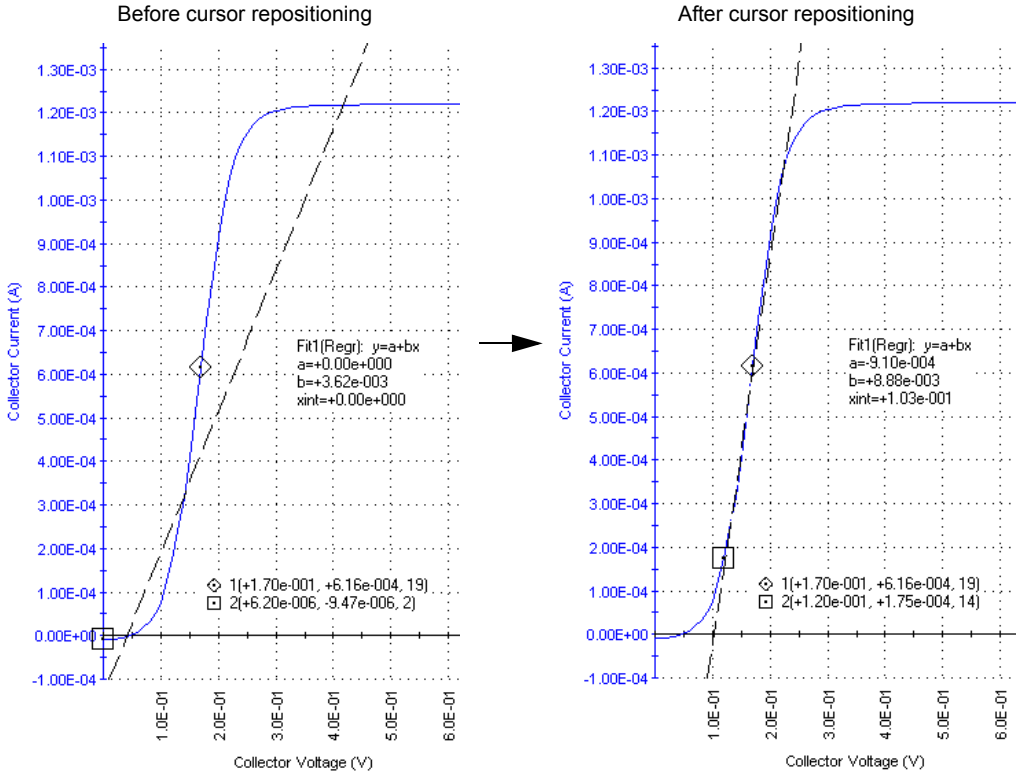
   Figure 6-313 illustrates the transition.

Figure 6-313
Result of checking the Fit #<CursorNumber> checkbox



7. If the cursors are not at the desired locations, then adjust them using the methods described under Positioning cursors on the graph using drag-and-drop or Positioning cursors on the graph using special options. See Figure 6-314.

Figure 6-314
Example result of repositioning a mating cursor



8. If you selected a **Tangent** fit AND you have no use for the extra cursor that was originally assigned to the fit, then do the following:
   a. Right-click the unwanted cursor.
   b. In the Cursor **<CursorNumber>** window, uncheck the **Visible** checkbox.
   c. Click **OK**. The Cursor **<CursorNumber>** window closes and the unwanted cursor disappears.

Numerically displaying extracted parameters and other data variables

On the graph, you can numerically display up to four values from the second row of a **Data** or **Calc** worksheet, along with the corresponding names from the first row. For example, you can display calculated, single-value extracted parameters, such as curve slopes, saturation values, and so on. In the **Data** worksheet, such values occupy the second row of an otherwise empty column.

For consistency with frequent industry usage, each second-row value (normally the first *data* value in a column [series] of data) will be referred to as a data variable in this subsection. Each first row value of a **Data** worksheet, and, ideally, the first-row value of a **Calc** worksheet, contains the name of the data variable.

If you select multiple data variables, all selected values are displayed together in a single text block, which may be located anywhere in the graph.

Selecting the data variables to be displayed

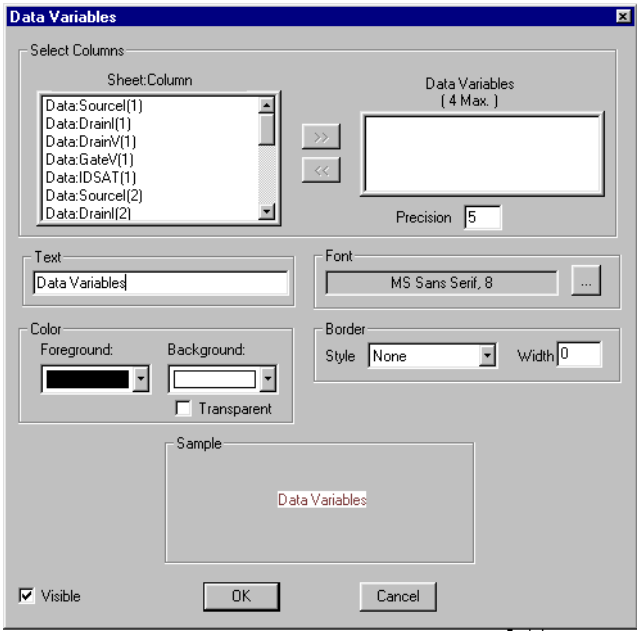
This subsection provides general instructions for selecting a data variable and presents a specific example involving an extracted **vds-id** parameter.

Instructions for selecting data variables

Select each data variable as follows:

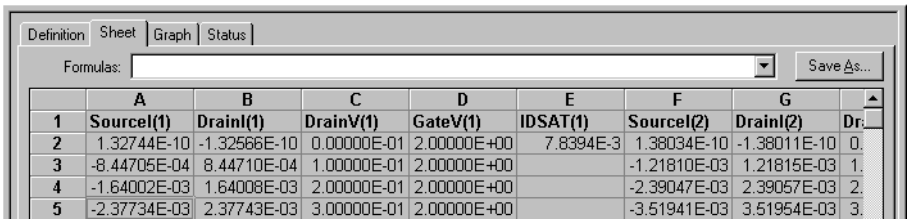
1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Data Variables.** The Data Variables window opens. See Figure 6-315.

Figure 6-315
Data Variables window



The data variable names that are displayed under **Sheet:Column** in the Data Variables window (Figure 6-315) are the same as the parameter names that are displayed in row 1 of the corresponding **Data** worksheet (Figure 6-316).

Figure 6-316
Data sheet displaying the data variable names shown in Figure 6-315



	A	B	C	D	E	F	G	
1	Sourcel(1)	Drainl(1)	DrainV(1)	GateV(1)	IDSAT(1)	Sourcel(2)	Drainl(2)	Dr
2	1.32744E-10	-1.32566E-10	0.00000E-01	2.00000E+00	7.8394E-3	1.38034E-10	-1.38011E-10	0.
3	-8.44705E-04	8.44710E-04	1.00000E-01	2.00000E+00		-1.21810E-03	1.21815E-03	1.
4	-1.64002E-03	1.64008E-03	2.00000E-01	2.00000E+00		-2.39047E-03	2.39057E-03	2.
5	-2.37734E-03	2.37743E-03	3.00000E-01	2.00000E+00		-3.51941E-03	3.51954E-03	3.

3. In the Data Variables window, under **Select Columns**, select up to four data variables that you want to display. Select the *names* of these data variables in the **Sheet:Column** list box, using one of the following methods:
   - **Method I**: Double-click a data variable name in the **Sheet:Column** list box. The data variable name is transferred to the **Data Variables** box.
   - **Method II**: 1) Select a data variable name in the **Sheet:Column** list box by single-clicking it; 2) click the **>>** button. The data variable name is transferred to the **Data Variables** box.

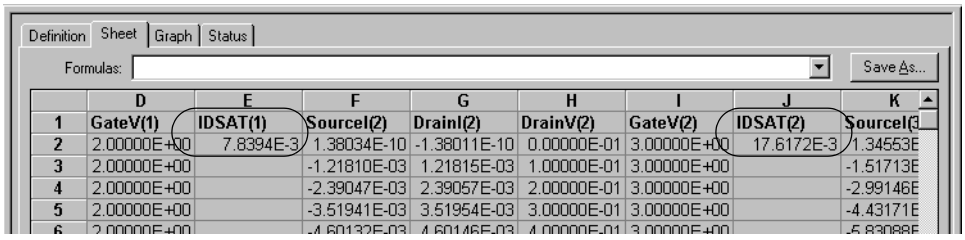
NOTE *In the **Sample** area of the Data Variable window, the selected data variables are displayed with values of zero. After you click **OK,** they are displayed on the graph with the actual values.*

4. If you wish to change any of the data variable selections in the Data Variable window, you can transfer a name from the **Data Variables** box back to the **Sheet:Column** list box, using one of the following methods:
   - **Method I**: Double-click the data variable name to be deleted from the **Data Variables** box. The data variable name is transferred back to the **Sheet:Column** list box.
   - **Method II**: 1) Select the data variable name to be deleted from the **Data Variables** box by single-clicking it; 2) click the **<<** button. The data variable name is transferred to the **Sheet:Column** list box.
5. By default, data variables display with a precision of five decimal places, indicated by the number 5 in the **Precision** text box (under the **Data Variables** box). If you prefer, enter a larger or smaller number of decimal places in the **Precision** text box.
6. Click **OK**. The selected data variables are displayed on the graph.

Example selection and display of data variables

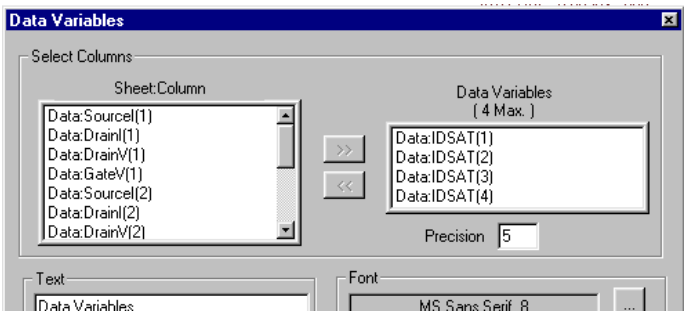
The as-shipped sample data for the **"vds-id" ITM** contains a series of **Formulator** calculated drain current saturation values, named **IDSAT**. See Figure 6-317.

Figure 6-317
IDSAT values for "vds-id" ITM



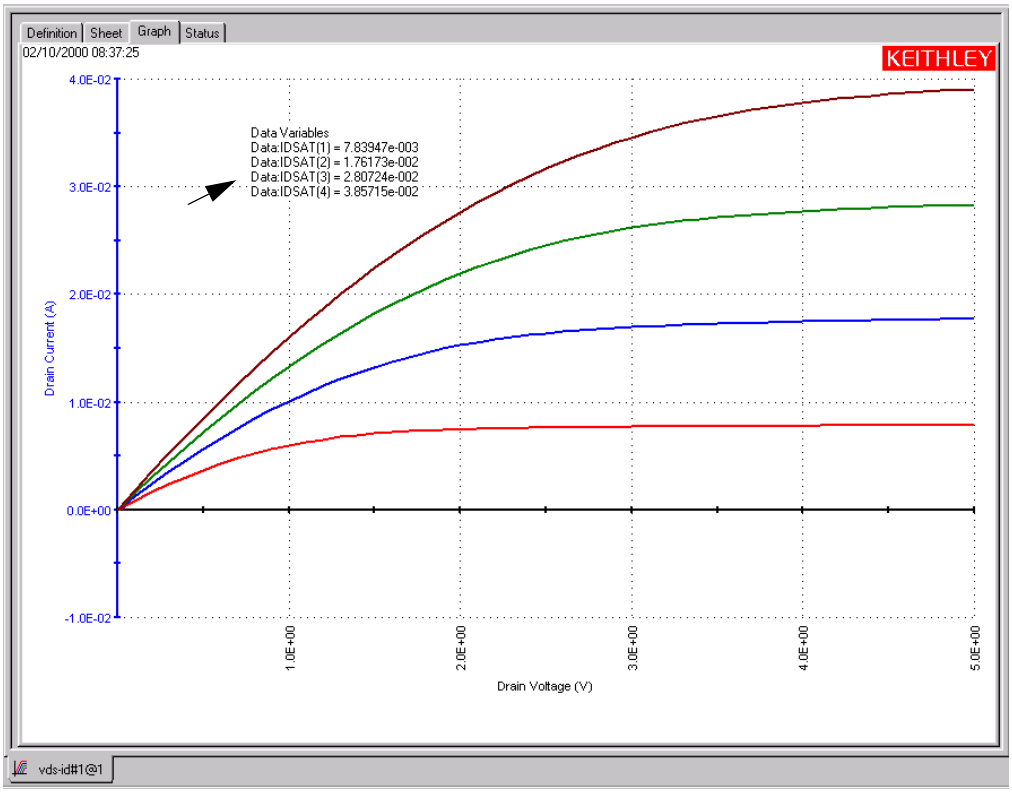
For illustration purposes, each of the four **IDSAT** values [**IDSAT(1)** through **IDSAT(4)**] was selected for display, using the Data Variable window. See Figure 6-318.

Figure 6-318
Selection of the four "vds-id" IDSAT values



As a result, the four **IDSAT** data variables were displayed on the graph, as shown in Figure 6-319.

Figure 6-319
Display of the four "vds-id" IDSAT data variables on the graph



Formatting the displayed data variables

This subsection explains how to change the heading, font, and color of the displayed data variables and how to place a frame around them.

Changing the heading that appears with the displayed data variables

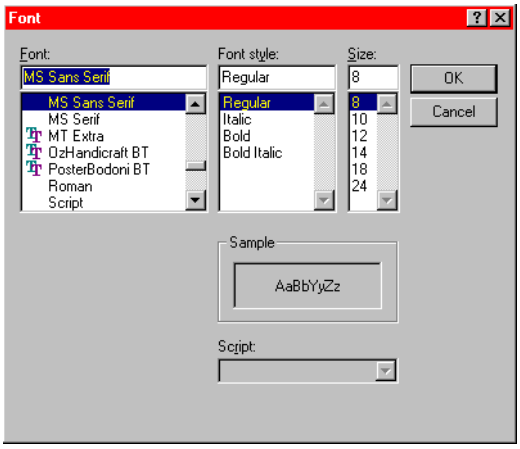
The displayed default name for the data variable text is **Data Variables**. To change this name, type a different name in the **Text** edit box of the Data Variables window. For example, for the **"vds-id"** graph above, one might choose to display the heading **Drain Saturation Current** over the four IDSAT values.

Changing the font of the displayed data variables

Change the data variable font as follows:

1. In the **Graph** tab Data Variables window, click the button at the right side of the **Font** combo box. The Font window appears. See Figure 6-320.

Figure 6-320
Font window



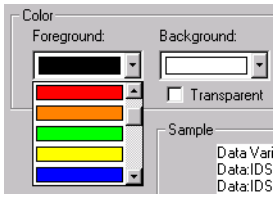
2. In the Font window, select the name, size, and style of the font desired for the data variable text. The **Sample** area of the Font window displays the new font.
3. Click **OK**. The **Sample** area of the Data Variables window displays the data variables with the selected font.
4. In the Data Variables window, click **OK**. The graph displays the data variables with the selected font.

Changing the text color of the displayed data variables

You can change the color of the data variable text, as follows:

1. Under **Color** in the **Graph** tab Data Variables window, click the scroll arrow on the **Foreground** combo box. Several color selections are available, some of which are shown in Figure 6-321.

Figure 6-321
Some of the available displayed data variable text colors



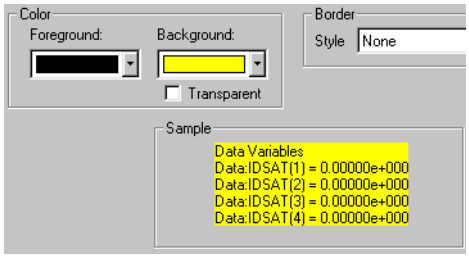
2. Select the desired text color. The **Sample** area displays the data variables text in the selected color.
3. Click **OK**. The new text color for the data variables is displayed on the graph.

Changing background color of the displayed data variables

You can change the background color for the displayed data variables as follows:

1. Under **Color** in the **Graph** tab Data Variables window, click the scroll arrow on the **Background** combo box. Several color selections are available; they are the same colors as for text color (for example, see Figure 6-321).
2. Select the desired text background color. The **Sample** area displays the background color. See Figure 6-322.

Figure 6-322
Example of data variable background color



3. Click **OK**. The new data variable background color is displayed in the graph.

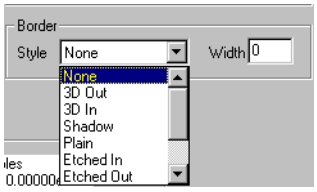
NOTE *If you want other graph components to be able to shine through the normally opaque background of the displayed data variables, select the **Transparent** checkbox on the **Color** area of the Data Variables window. However, be aware that a checked **Transparent** checkbox 1) overrides the background color selection (signified by a gray color in the **Background** combo box); and 2) causes a border to be displayed in gray scale (for more about borders, refer to the following paragraph).*

Adding/changing a border around the displayed data variables

You can add or change a border around the displayed data variables, as follows:

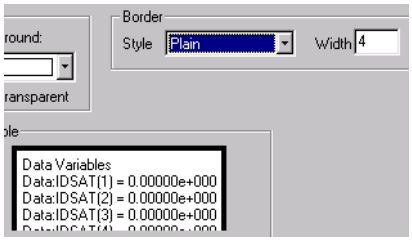
1. Under **Border** in the **Graph** tab Data Variables window, click the scroll arrow on the **Style** combo box. Several border style selections are available, some of which are shown in Figure 6-323.

Figure 6-323
Some of the available borders for displayed data variables



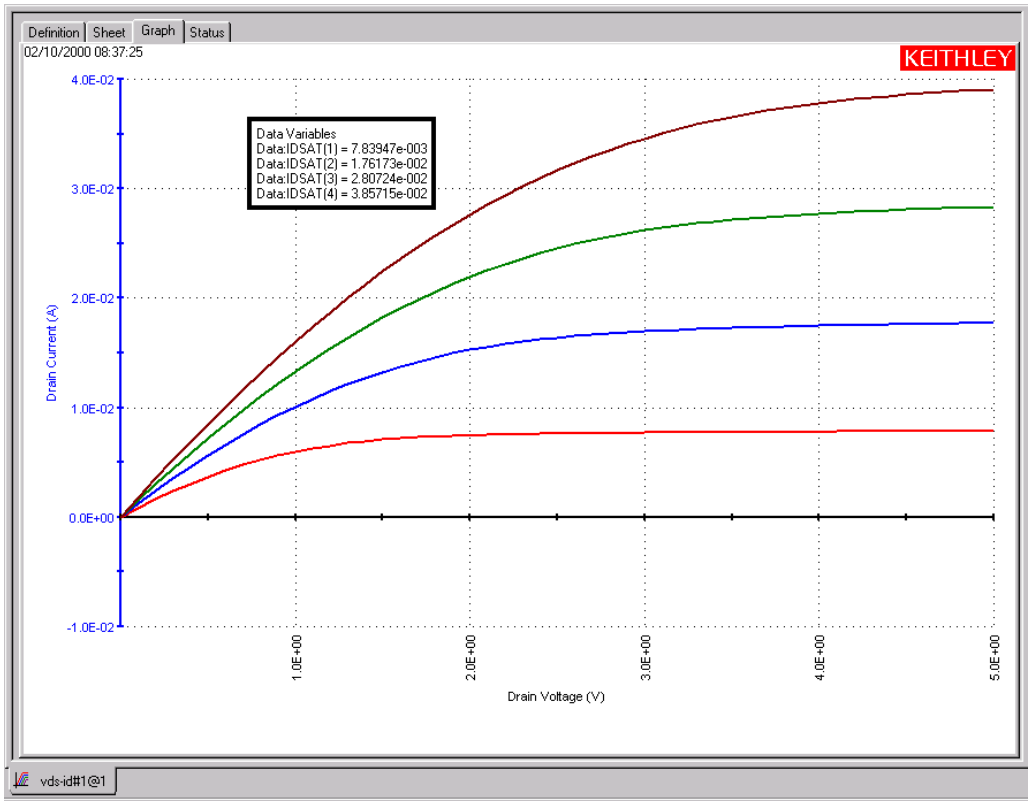
2. Select the desired border style (Note that if the adjacent **Width** setting is 0 [default], the **Sample** area does not yet display a border).
3. Also under **Border** in the Data Variables window, in the **Width** text box, type a width for the border. KITE accepts values between 0 and 20 pixels. The **Display Sample** area displays the selected border around the values, in the same color as selected in the **Background** combo box. See Figure 6-324.

Figure 6-324
Example 4-pixel Plain border for displayed data variables



4. Click **OK**. The graph displays the data variables with the selected border. Figure 6-325 shows a bordered display of vds-id data variables in the default position on the graph.

Figure 6-325
Display of Plain-bordered "vds-id" data variables



Positioning the displayed data variables

The displayed data variables can be positioned anywhere on the graph, as follows:

1. On the graph, click on the displayed data variables with the left button of the mouse or other pointing device and continue holding the button down.
2. Drag the displayed data variables to the desired position on the graph.
3. Release the pointing device button.

Hiding the data variables

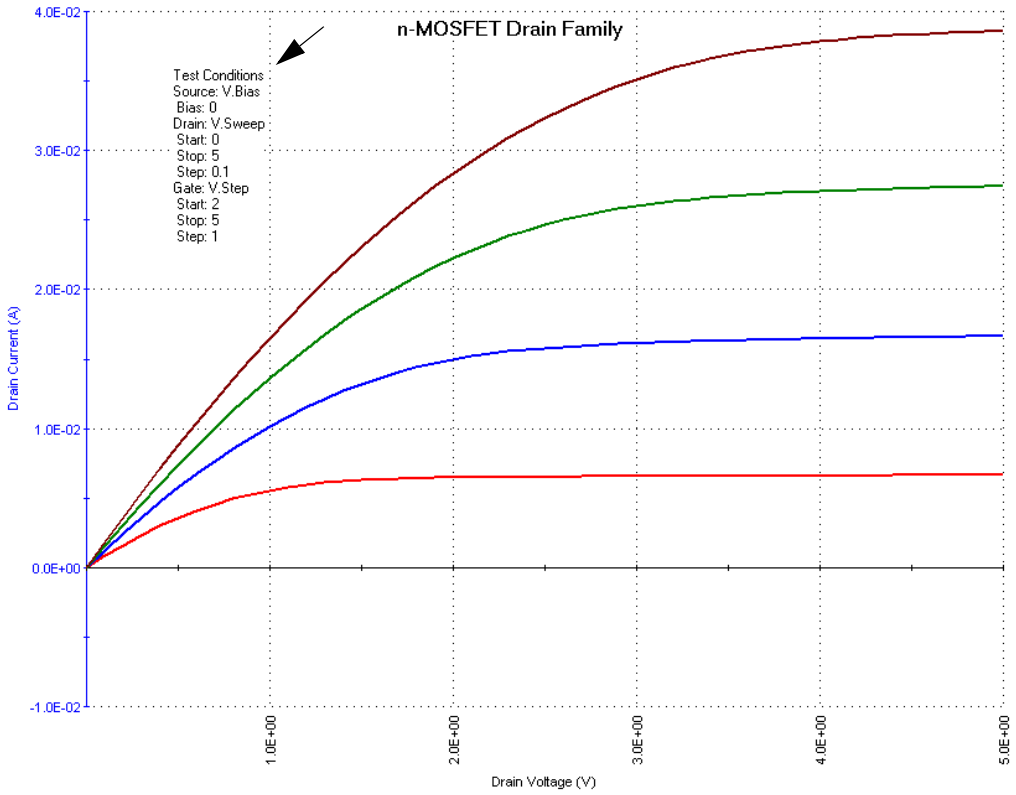
You can hide the data variables from the graph by unchecking the **Visible** checkbox of the Data Variables window or by unchecking the **Data Variables** item in the **Graph Settings** menu and pressing **OK**. The other settings of the Data Variables window are retained.

To restore display of the data variables to the graph, check the **Visible** checkbox of the Data Variables window and press **OK**. Alternatively, in the **Graph Settings** menu, check the **Data Variables** item by clicking it.

Displaying test conditions

Clicking the **Test Conditions** item in the **Graph Settings** menu displays the primary test conditions used to obtain the data in a graph. See Figure 6-326.

Figure 6-326
Test-conditions display for a "vds-id" graph



Understanding which test conditions are displayed

Table 6-12 lists the test conditions that **Graph Settings** → **Test Conditions** displays for the device under test (DUT).

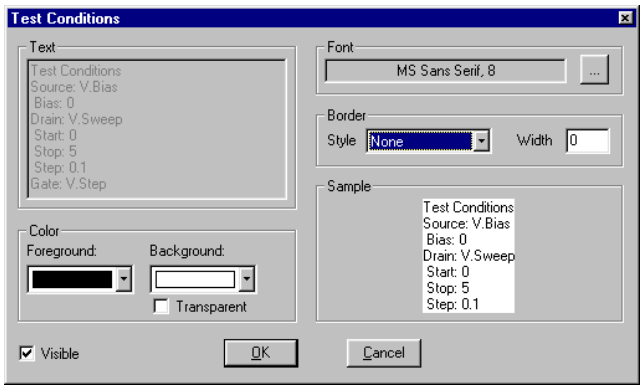
Table 6-13
Test Conditions information displayed for the terminals of the DUT

For each terminal of the DUT, the Test Conditions menu item displays the name, the applied forcing function, and the corresponding test conditions as listed below.	
Forcing function	**Listed test conditions**
Open	None
Common	None
GNDU	None
Bias	**Level** value
Sweep, linear mode	**Start** value
	Stop value
	Step value
Sweep, log mode	**Start** value
	Stop value
	Data Points value
List sweep	**Data Points** value
Step	**Start** value
	Stop value
	Step value

Formatting the displayed test conditions

You can frame the displayed test conditions and format the font and color using the Test Conditions window. Open the Test Conditions window by right-clicking the displayed test conditions. See Figure 6-327.

Figure 6-327
Test Conditions window

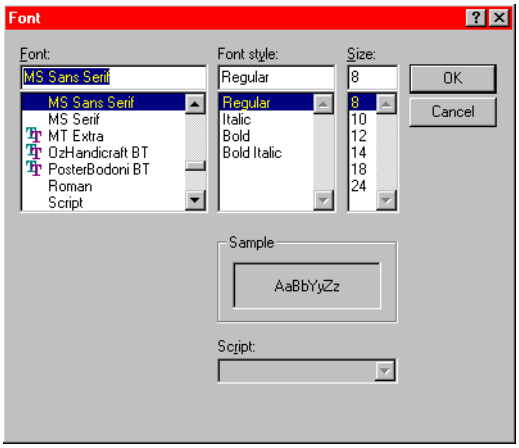


Changing the font of the displayed test conditions

Change the test conditions font as follows:

1. In the **Graph** tab Test Conditions window, click the button at the right side of the **Font** combo box. The Font window appears. See Figure 6-328.

Figure 6-328
Font window



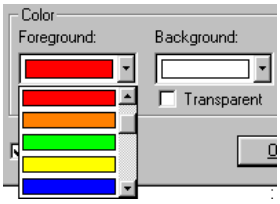
2. In the Font window, select the name, size, and style of the font desired for the test conditions text. The **Sample** area of the Font window displays the new font.
3. Click **OK**. The **Sample** area of the Test Conditions window displays the test conditions with the selected font.
4. In the Test Conditions window, click **OK**. The graph displays the test conditions with the selected font.

Changing the text color of the displayed test conditions

Change the color of the test condition text as follows:

1. Under **Color** in the **Graph** tab Test Conditions window, click the scroll arrow on the **Foreground** combo box. Several color selections are available, some of which are shown in Figure 6-329.

Figure 6-329
Some of the available displayed test-condition text colors



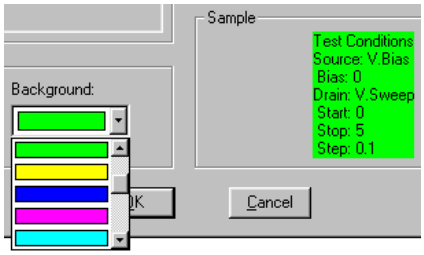
2. Select the desired text color. The **Sample** area displays the test conditions text in the selected color.
3. Click **OK**. The new text color for the test conditions is displayed on the graph.

Changing background color of the displayed test conditions

Change the background color for the displayed test conditions as follows:

1. Under **Color** in the **Graph** tab Test Conditions window, click the scroll arrow on the **Background** combo box. Several color selections are available; they are the same colors as for text color.
2. Select the desired text background color. The **Sample** area displays the background color. See Figure 6-330.

Figure 6-330
Example of test condition background color



3. Click OK. The new test conditions background color is displayed in the graph.

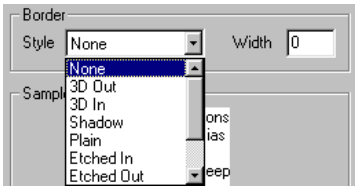
NOTE *If you want other graph components to be able to shine through the normally opaque background of the displayed test conditions, select the **Transparent** checkbox on the **Color** area of the Test Conditions window. However, be aware that a checked **Transparent** checkbox 1) overrides the background color selection (signified by a gray color in the **Background** combo box); and 2) causes a border to be displayed in gray scale (for more about borders, refer to the following paragraph).*

Adding/changing a border around the displayed test conditions

You can add or change a border around the displayed test conditions, as follows:

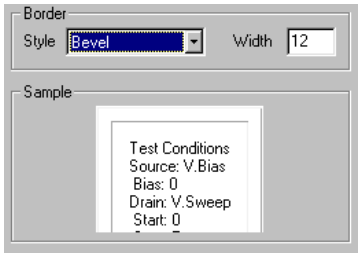
1. Under **Border** in the **Graph** tab Test Conditions window, click the scroll arrow on the **Style** combo box. Several border style selections are available, some of which are shown in Figure 6-331.

Figure 6-331
Some of the available borders for displayed test conditions



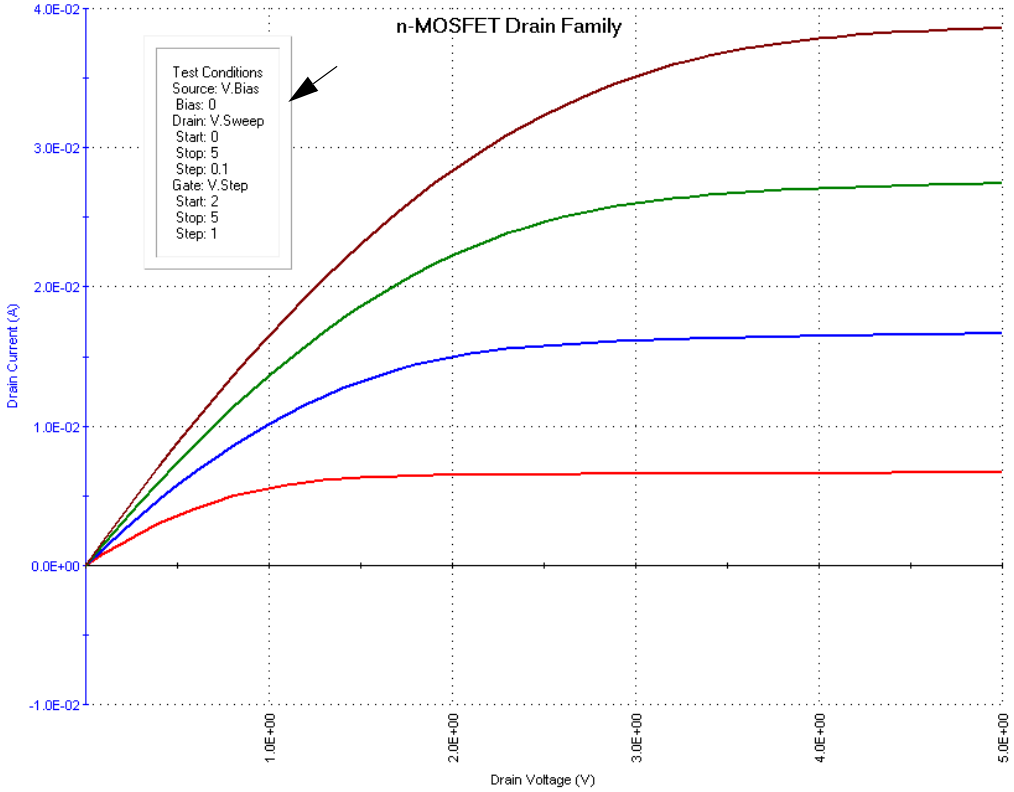
2. Select the desired border style (note that if the adjacent **Width** setting is 0 [default], the **Sample** area does not yet display a border).
3. Also under **Border** in the Test Conditions window, in the **Width** text box, type a width for the border. KITE accepts values between 0 and 20 pixels. The **Sample** area displays the selected border around the values, in the same color as selected in the **Background** combo box. See Figure 6-332.

Figure 6-332
Example 12-pixel Bevel border for displayed test conditions



4. Click OK. The graph displays the test conditions with the selected border. Figure 6-333 shows a bordered display of **"vds-id"** test conditions on the graph.

Figure 6-333
Display of Bevel-bordered "vds-id" test conditions



Positioning the displayed test conditions

The displayed test conditions can be positioned anywhere on the graph, as follows:

1. On the graph, click on the displayed test conditions with the left button of the mouse or other pointing device and continue holding the button down.
2. Drag the displayed test conditions to the desired position on the graph.
3. Release the pointing device button.

Hiding the test conditions

You can hide the test conditions from the graph by unchecking the **Visible** checkbox of the Test Conditions window or by unchecking the **Test Conditions** item in the **Graph Settings** menu and pressing **OK**. The other settings of the Test Conditions window are retained.

To restore display of the test conditions to the graph, check the **Visible** checkbox of the Test Conditions window and press **OK**. Alternatively, in the **Graph Settings** menu, check the **Test Conditions** item by clicking it.

Adding a title, legend, or comment to the graph

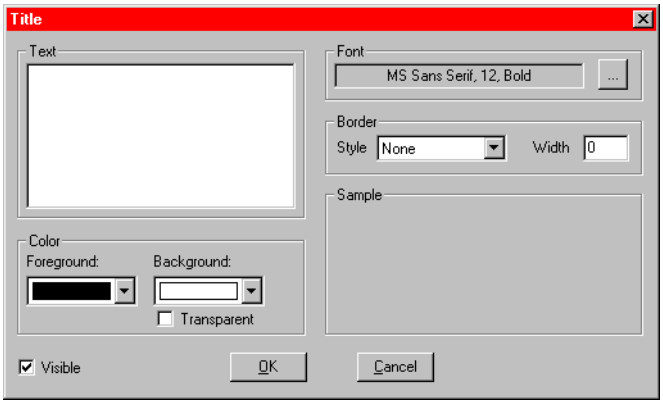
This subsection discusses addition of a title, legend, or comment *individually*. However, the subsection discusses formatting and positioning of these components *collectively*, because the approach is essentially identical for all cases.

Adding a title

Add a title to the graph as follows:

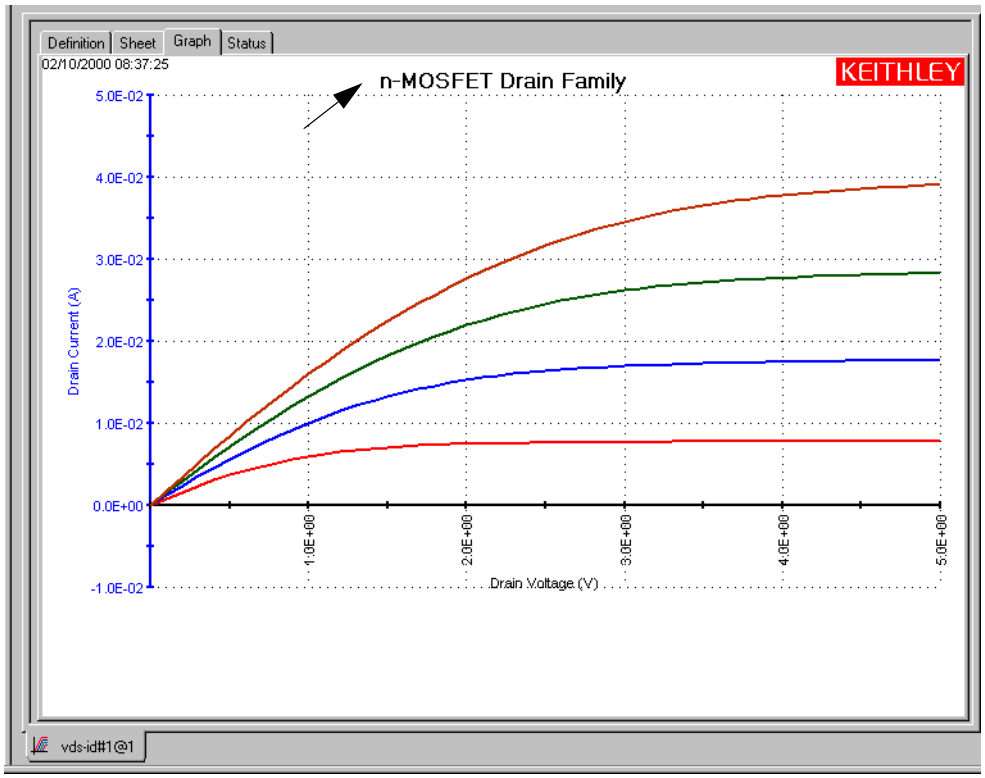
1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Title.** The Title window opens. See Figure 6-334.

Figure 6-334
Title window



3. In the Title window, under **Text**, type the desired title for your graph. The title displays in the **Sample** area with the selected font, text and background color, and border (refer to Formatting a title, legend, or comment).
4. Click **OK**. The title displays in the default position on the graph. Figure 6-335 shows a title for the **vds-id ITM**.

Figure 6-335
Display of a graph title



5. If desired, reposition the displayed title as follows:
   a. On the graph, click on the displayed title with the left button of the mouse or other pointing device and continue holding the button down.
   b. Drag the displayed title to the desired position on the graph.
   c. Release the pointing device button.

Adding a legend

Add a legend to the graph as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Legend.** The legend displays on the graph, automatically showing the previously selected plot formats (refer to Defining the plot properties of the graph: colors, line patterns, symbols, line widths).

   Figure 6-336 shows the displayed legend for the color-coded vds-id graph, in the default position. Figure 6-337 shows the legends for three other plot codings, as follows:
   - **Uncoded**: The default. To code multiple plots, use one of the available methods described under Defining the plot properties of the graph: colors, line patterns, symbols, line widths.
   - **Line format coded**: The legend for the plot shown in Figure 6-267.
   - **Plot symbol coded**: The legend for the plot shown in Figure 6-268.

Figure 6-336
Display of a legend for color coded plots

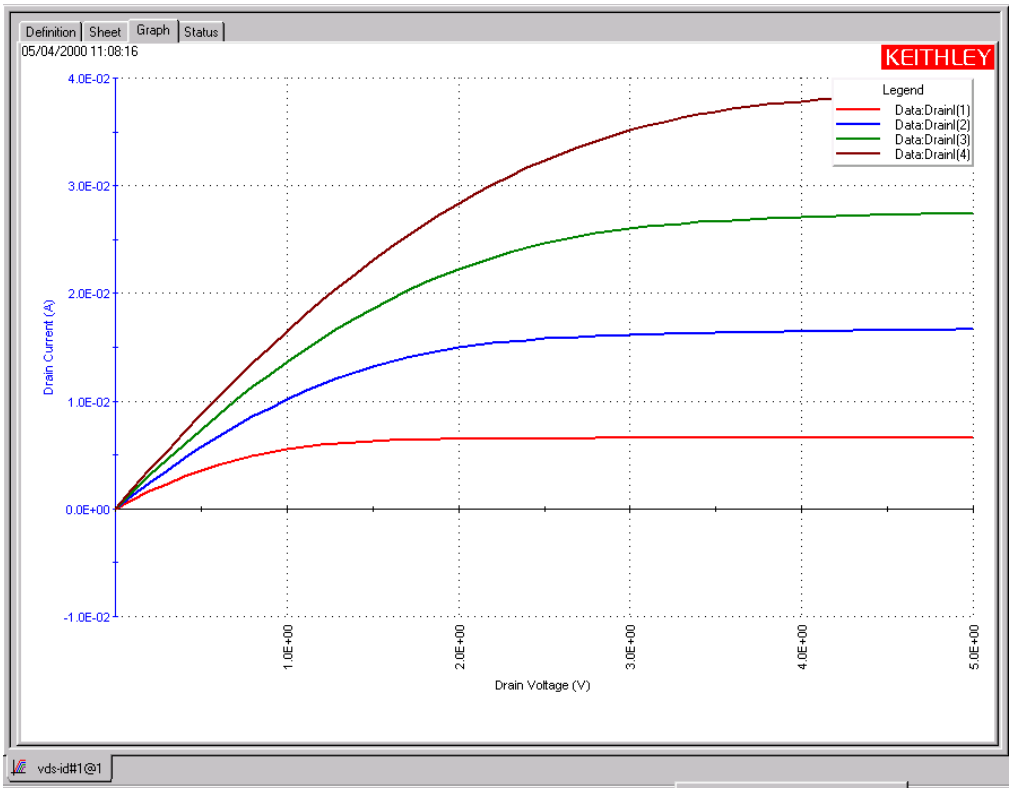
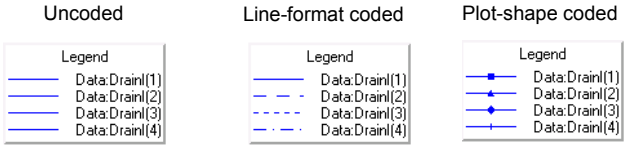


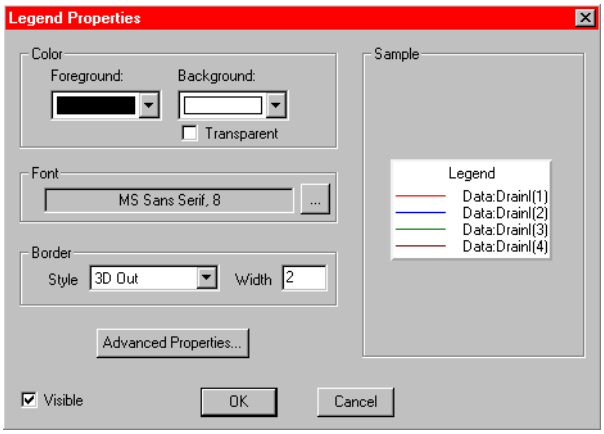
Figure 6-337
Display of legends for uncoded, line-format coded, and plot-symbol coded plots



3. If you wish to reformat the font, text, or background color border of the legend or series names, open the Legend Properties window by either of the following methods:
   - On the graph, right-click on the legend.
   - In the **Graph Settings** menu, select **Graph Properties** → **Legend**.

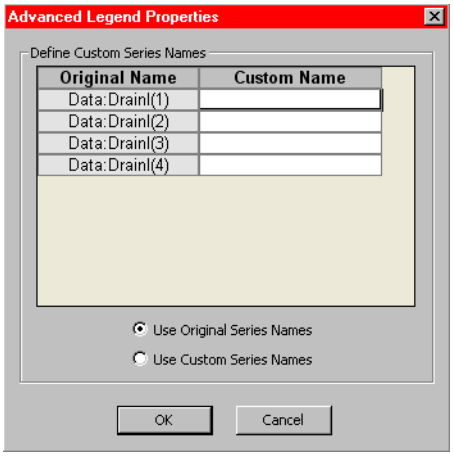
   The Legend Properties window opens. See Figure 6-338.

Figure 6-338
Legend Properties window



4. The legend displays in the **Sample** area with the current font, text and background color, and border. To change these, refer to Formatting a title, legend, or comment.
5. **Advanced Properties**: By default, the series names for the graph are the same as the names that appear on the spreadsheet. If desired, you can change the series name(s) for the graph legend as follows:
   a. Click **Advanced Properties** to display the **Advanced Legend Properties** (see Figure 6-340).
   b. The **Original Name** for each series is listed in the first column of the chart.
   c. To define a **Custom Name**, click the text box next to the original name, and type in the desired name.
   d. To use the custom names in the legend, select **Use Custom Series Names**.
   e. Click **OK** to close the properties window.
6. If desired, reposition the displayed legend as follows:
   a. On the graph, click on the displayed legend with the left button of the mouse or other pointing device and continue holding the button down.
   b. Drag the displayed legend to the desired position on the graph.
   c. Release the pointing device button.

Figure 6-339
Advanced legend properties

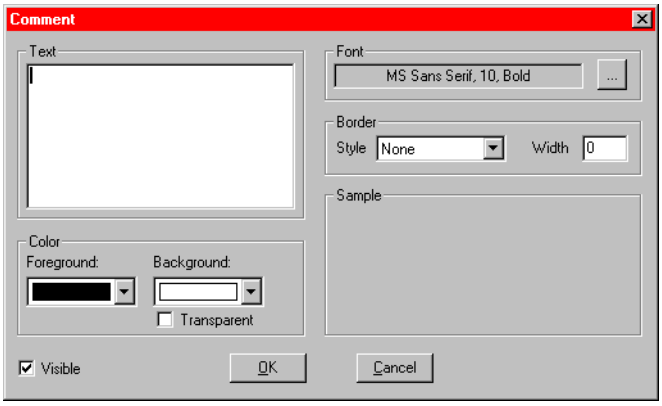


Adding a comment

Add a comment to the graph as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Comment.** The Comment window opens. See Figure 6-340.

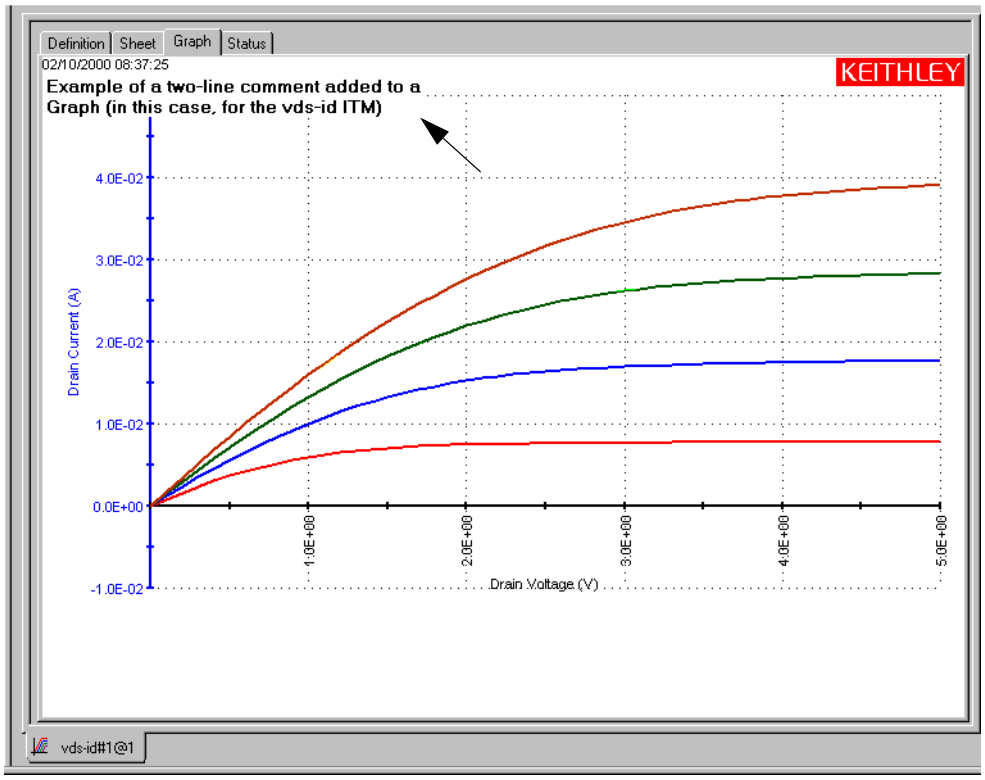
Figure 6-340
Comment window



3. In the Comment window, under **Text**, type the desired comment for your graph. The comment displays in the **Sample** area (sometimes partially, depending on length) with the selected font, text and background color, and border (refer to Formatting a title, legend, or comment).

4. Click **OK**. The comment displays on the graph. See the example in Figure 6-341.

Figure 6-341
Display of a graph comment in default position



5. If desired, reposition the displayed comment as follows:
   a. On the graph, click on the displayed comment with the left button of the mouse or other pointing device and continue holding the button down.
   b. Drag the displayed comment to the desired position on the graph.
   c. Release the pointing device button.

Formatting a title, legend, or comment

Change the font, text or background color, or border of a title, legend, or comment in essentially the same way as you format cursor coordinates and data variables and test conditions. Open the Title, Legend Properties, or Comment window, as appropriate by either of the following:

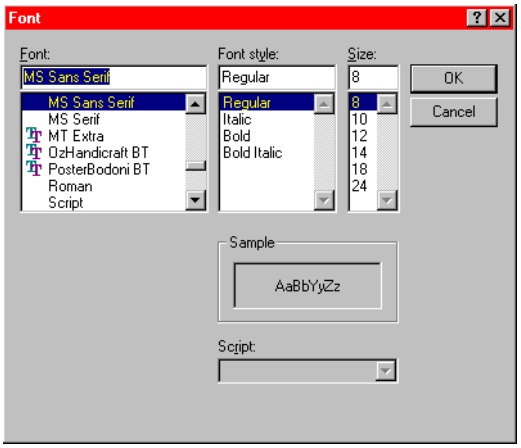
- On the graph, right-click on the title, legend, or comment.
- In the **Graph Settings** menu, select **Graph Properties** → **Title**, **Graph Properties** → **Legend**, or **Graph Properties** → **Comment**, as appropriate.

Changing the font of a title, legend, or comment

Change the font as follows:

1. In the **Graph** tab Title, Legend Properties, or Comment window, as appropriate, click the button at the right side of the **Font** combo box. The Font window appears. See Figure 6-342.

Figure 6-342
Font window



2. In the Font window, select the name, size, and style of the desired font for the title, legend, or comment. The **Sample** area of the Font window displays the new font.
3. Click **OK**. The **Sample** area of the Title, Legend Properties, or Comment window displays the title, legend, or comment with the selected font.
4. In the Title, Legend Properties, or Comment window, click **OK**. The graph displays the title, legend, or comment with the selected font.

Changing the text color of a title, legend, or comment

You can change the text color as follows:

1. Under **Color** in the **Graph** tab Title, Legend Properties, or Comment window, as appropriate, click the scroll arrow on the **Foreground** combo box. Several color selections are available, some of which are shown in Figure 6-343.

Figure 6-343
Some of the available displayed text colors for a title, legend, or comment

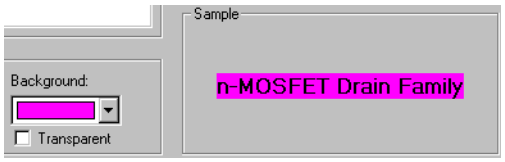
2. Select the desired text color. The **Sample** area of the Title, Legend Properties, or Comment window displays the text in the selected color.
3. Click **OK**. The new text color for the title, legend, or comment is displayed on the graph.

Changing the background color of a title, legend, or comment

You can change the text background color as follows:

1. Under **Color** in the **Graph** tab Title, Legend Properties, or Comment window, as appropriate, click the scroll arrow on the **Background** combo box. Several color selections are available; they are the same colors as for text color (for example, see Figure 6-343).
2. Select the desired text background color. The **Sample** area displays the background color of the title, legend, or comment. See Figure 6-344.

Figure 6-344
Example of background color for a title



3. Click **OK**. The new background color is displayed in the graph.

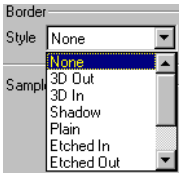
NOTE *If you want other graph components to be able to shine through the normally opaque background of the displayed title, legend, or comment, then select the **Transparent** checkbox on the **Color** area of the Title, Legend Properties, or Comment window. However, be aware that a checked **Transparent** checkbox 1) overrides the background color selection (signified by a gray color in the **Background** combo box); and 2) causes a border to be displayed in gray scale (for more about borders, refer to the next subsection).*

Adding/changing a border around a title, legend, or comment

You can add or change a border around the title, legend, or comment as follows:

1. Under **Border** in the **Graph** tab Title, Legend Properties, or Comment window, as appropriate, click the scroll arrow on the **Style** combo box. Several border style selections are available, some of which are shown in Figure 6-345.

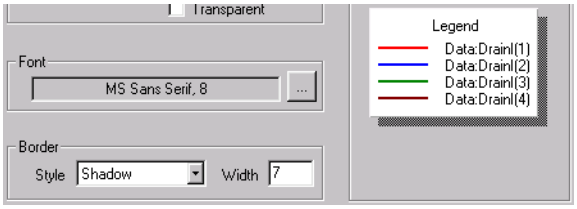
Figure 6-345
Some of the available borders for a title, legend, or comment



2. Select the desired border style (note that if the adjacent **Width** setting is 0 [default], the **Sample** area does not yet display a border).
3. Also under **Border** in the Title, Legend Properties, or Comment window, in the **Width** test box, type a width for the border. KITE accepts values between 0 and 20 pixels. The **Display Sample** area displays the selected border around the title, legend, or comment, in the same color as selected in the **Background** combo box.

   Figure 6-346 shows an example border, in this case around a legend (a legend displays with a border by default. Figure 6-336 shows a different border).

Figure 6-346
Example of 7-pixel Shadow legend border displayed in Sample area



4. Click **OK**. The graph displays the title, legend, or comment with the selected border.

Hiding a title, legend, or comment

You can hide a title, legend, or comment from the graph by unchecking the **Visible** checkbox of the Title, Legend Properties, or Comment window or by unchecking the **Title**, **Legend**, or **Comment** item in the **Graph Settings** menu. Hiding the component does not change its other settings.

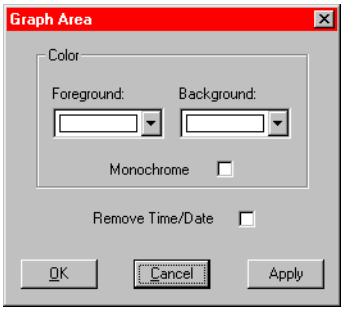
To restore display of the title, legend, or comment to the graph, check the **Visible** checkbox of the Title, Legend Properties, or Comment window. Alternatively, in the **Graph Settings** menu, check the **Title**, **Legend**, or **Comment**, as appropriate, by clicking it.

Changing area properties of the graph

You can change the colors of graph foreground (the plot area) and background (outside the plot area). You can also make everything on the graph monochrome and hide the time and date display. To change these properties, open the Graph Area window as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Graph Properties → Graph Area**. The Graph Area window opens. See Figure 6-347.

Figure 6-347
Graph Area menu

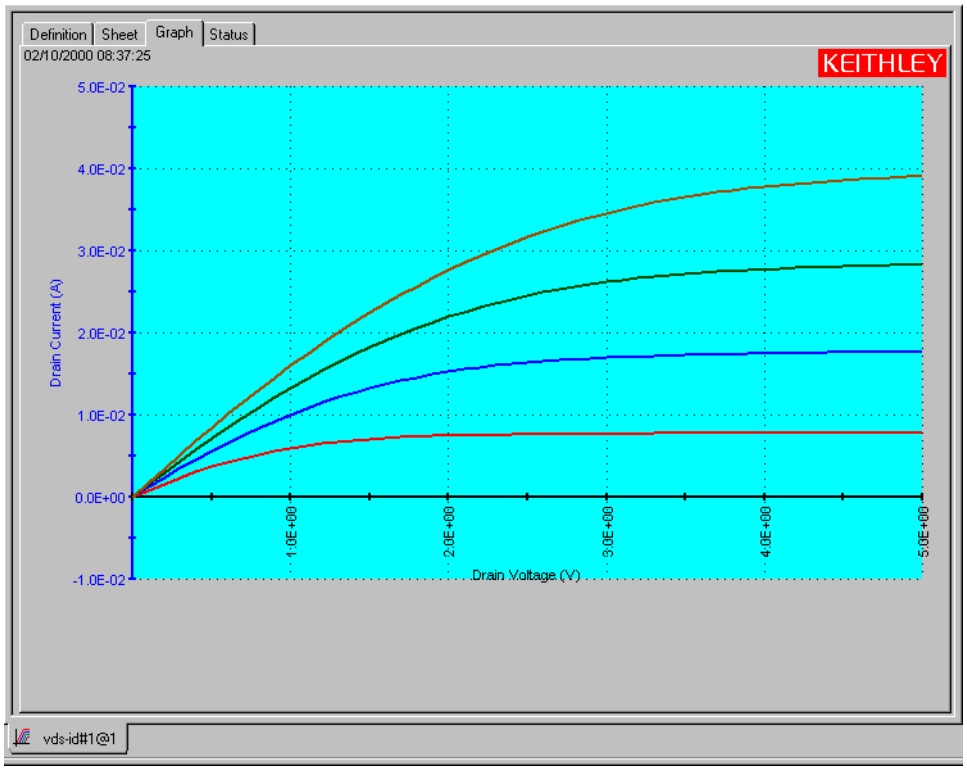


Changing graph foreground and background colors.

Select the colors of graph foreground (the plot area) and background (outside the plot area) using the **Foreground** and **Background** combo boxes in the Graph Area window (the available colors are identical to the available colors for all other **Graph** tab components). Then click **Apply**. The new colors are applied to the graph.

Figure 6-348 shows the vds-id graph with some contrasting foreground and background colors.

Figure 6-348
Illustration of foreground and background colors



Creating a monochrome graph

CAUTION **Unchecking the Monochrome checkbox and then clicking Apply does *not* restore the pre-monochrome colors.**

If you check the **Monochrome** checkbox, by clicking it, and then click **Apply**, the entire graph displays in black and white. The foreground and background color selection is overridden, and the foreground and background colors are both white. The line colors change to black and white, and the line patterns are automatically selected.

Hiding the displayed date and time

To hide the displayed date and time, do the following:

1. In the Graph Area window, check the **Remove Time/Date** checkbox by clicking it.
2. Click **Apply**. The date and time display disappears from the graph.

To restore the displayed date and time, do the following:

1. In the Graph Area window, uncheck the **Remove Time/Date** checkbox by clicking it.

2. Click **Apply**. The date and time display reappears on the graph.

Changing the size of a graph

Temporarily enlarging a selected area of the graph by zooming

You can enlarge and examine a small part of the graph for viewing (only) by zooming-in. The axis scales adjust automatically, independently of the autoscaling setting. By sequentially zooming-in multiple times, you can observe and quantify a very small portion of the graph.

NOTE *Zooms are temporary characteristics of the graph and cannot be saved.*

Zooming-in

To zoom-in (enlarge a part of the graph), do the following:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Zoom In.** The cursor changes to a tiny magnifying glass.
3. Using the magnifying glass cursor, draw a rectangle that defines the area of the graph to be enlarged, as follows:
   a. Place the center of the cursor at one corner of the area to be enlarged.
   b. Press down and hold the left button of the pointing device (mouse).
   c. Pull the cursor diagonally to the opposite corner of the area to be enlarged; this action displays a rectangular frame around the area.
   d. Release the left button of the pointing device (mouse).

   The part of the graph that you selected is now enlarged to occupy the full graph area.

The rectangle in Figure 6-349 shows a selected area of the **"vds-id"** graph to be enlarged. Figure 6-350 shows the selected area after enlargement.

Figure 6-349
Area to be enlarged by Zoom In

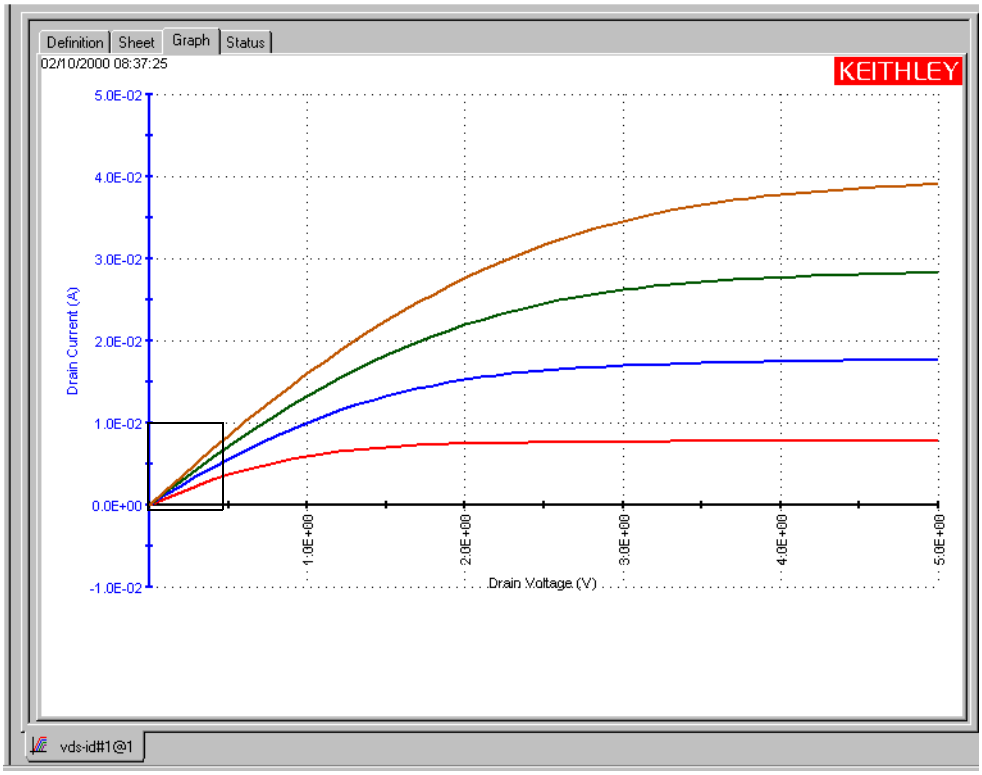
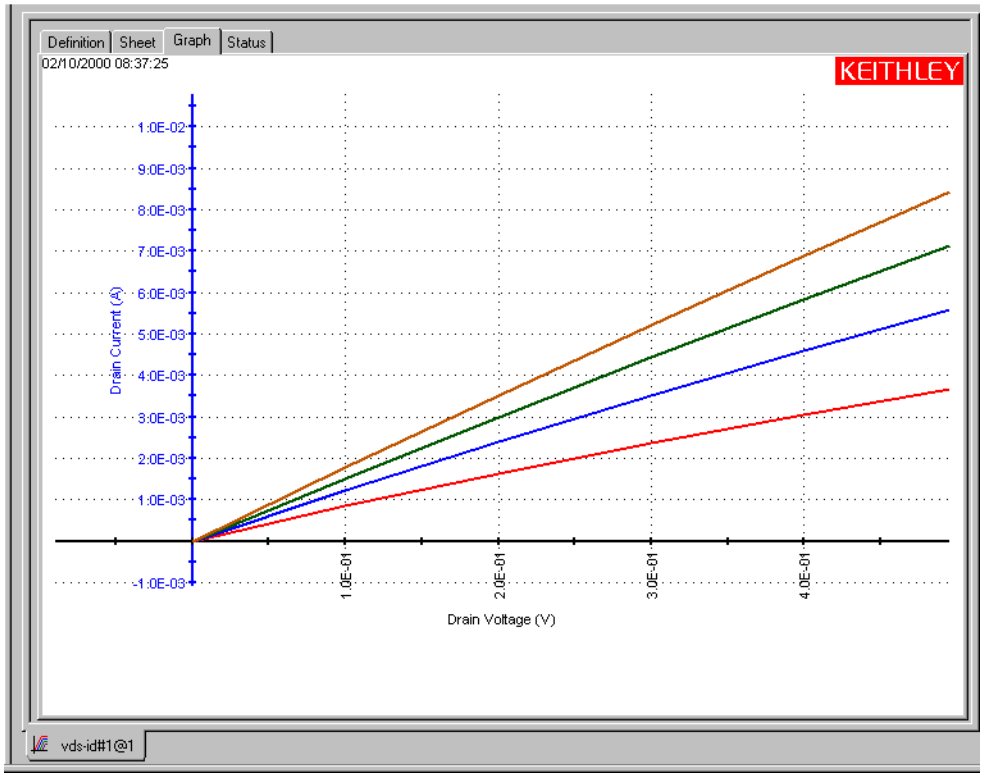


Figure 6-350
Selected area of Figure 6-349 after enlargement by Zoom In



NOTE *An increased axis label precision should be specified when examining very small areas of the graph (not essential in the case of Figure 6-350). Refer to Customizing axis locations and formats.*

Zooming-out

To restore a graph to the original or previously zoomed size, use the **Zoom Out** menu selection, as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Zoom Out**. The graph is restored to the original size, or to the previously zoomed size if multiple zooms were involved.

Changing the size property of the graph

You can increase or decrease the size of a graph such that, unlike zooming, the new size is a property of the graph (the new size is saved when the graph is saved). Do this as follows:

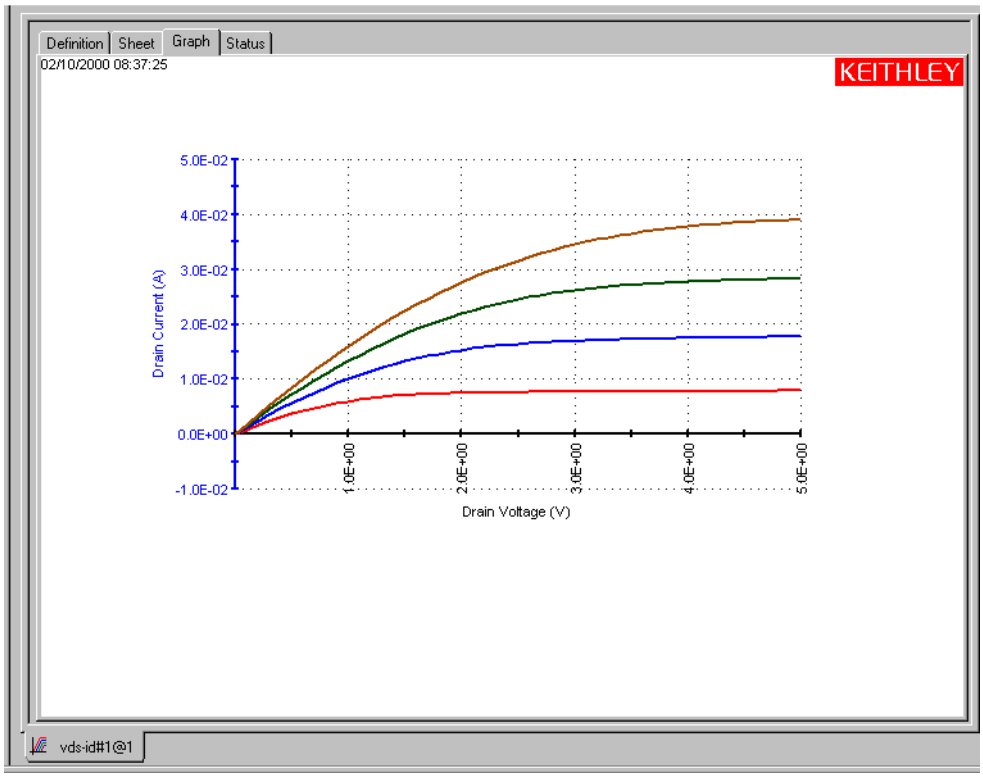
1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Resize.** The cursor changes to a diagonal ruler, and a check (√) appears next to the **Resize** menu entry, indicating that the graph is in **Resize** mode.
3. Resize the graph by moving the ruler diagonally.
4. When you have finished resizing the graph, repeat steps 1 and 2 to end the **Resize** mode and restore the cursor to normal (the **Resize** menu selection toggles).

NOTE *A resized graph remains centered on the **Graph** tab.*

*A size change through the **Resize** menu selection is saved when you save the graph (by contrast, a size change through the **Zoom In** or **Zoom Out** menu selection is temporary and is ignored when you save the graph).*

Figure 6-351 shows a downsized **"vds-id"** graph.

Figure 6-351
Resized graph example



Changing the position of a graph

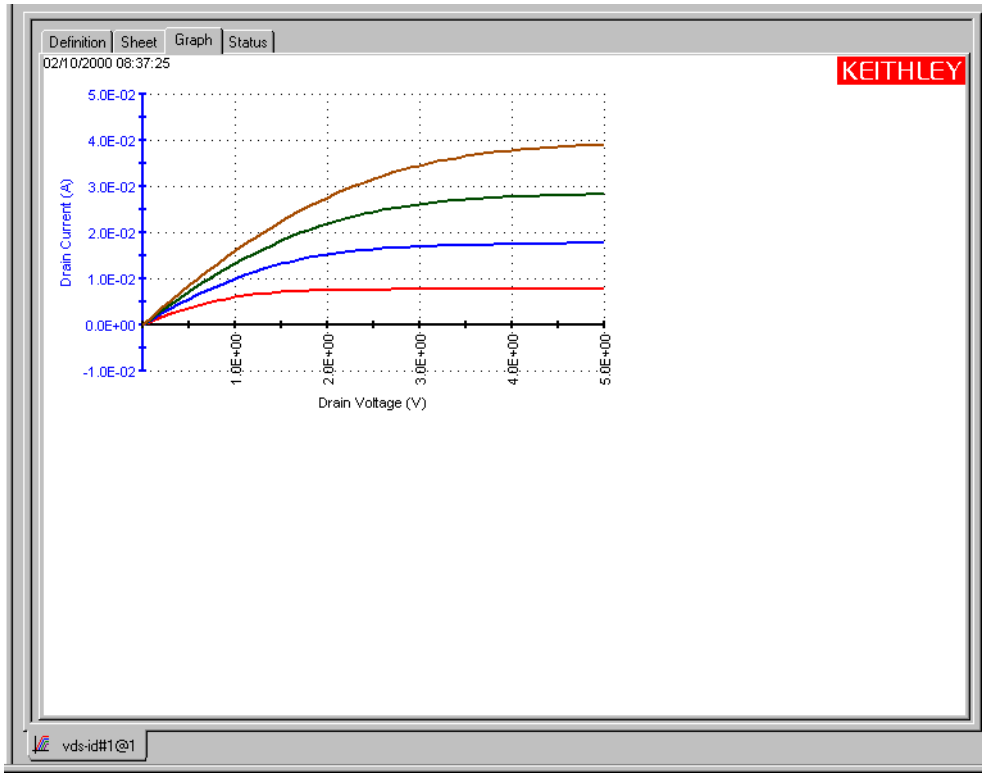
You can reposition a graph on the **Graph** tab, as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Move.** The cursor changes to crossed arrows, and a check (√) appears next to the **Move** menu entry, indicating that the graph is in **Move** mode.
3. Move the graph by moving the cursor.
4. When you have finished repositioning the graph, repeat steps 1 and 2 to end the **Move** mode and restore the cursor to normal (the **Move** menu selection toggles).

NOTE *A change through the **Move** menu selection is saved when you save the graph.*

Figure 6-352 shows a downsized **vds-id** graph that has been repositioned using the **Move** menu selection.

Figure 6-352
Resized and repositioned graph example



Identically configuring the graphs resulting from one test executed at multiple sites

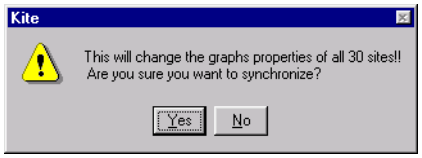
Ideally, data from a single ITM or UTM instance that is performed at multiple, identically fabricated sites should be plotted on multiple, identically configured graphs. Manual matching of graph configurations for multiple sites is time-consuming, tedious, and potentially prohibitive if the number of sites is large. However, for a single test instance in the project plan, you can use the KITE **Synchronize Graphs** function to *automatically* configure the graphs identically for all sites using one of the graphs as a master. For example, if you executed the vds-id#1 ITM at the first five sites of a project, **Synchronize Graphs** identically configures graphs for the following **Data** worksheets: vds-id#1@1, vds#1@2, vds-id#1@3, vds#1@4, and vds-id#1@5.

NOTE *If the project plan contains multiple instances of a same-named test, you must apply the feature separately each such instance. For example, if the Project Navigator shows both vds-id#1 and vds-id#2 ITMs, you must apply* ***Synchronize Graphs*** *separately for vds-id#1 and vds-id#2.*

Identically configure the multi-site graphs as follows:

1. Open the **Graph** tab for a selected master site as described below:
   a. Using the scroll bar of the Site Navigator, select the site for which you want to configure a master graph.
   b. In the Project Navigator, double-click the ITM or UTM for which the data is to be graphed. The corresponding ITM or UTM window opens.
   c. In the ITM or UTM window, open the **Graph** tab for the ITM or UTM.
2. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
3. In the **Graph Settings** menu, select **Synchronize Graphs.** A caution message appears. See Figure 6-353.

Figure 6-353
Caution message for the Synchronize Graphs feature



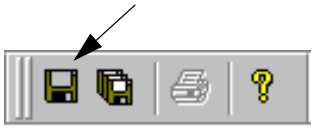
CAUTION **The graphs for the selected test instance will be configured identically for all project sites, both for the present data *and for all future data.* This applies to future graphs for *all sites, even if data was not yet generated for some sites at the time Synchronize Graphs was requested.* The only way to undo these effects is to manually reconfigure site specific graphs individually.**

4. If you are sure that you wish to proceed, click **Yes**. The graphs for the selected test are now configured identically for all project sites.

Saving a graph

Saving the graph to the project

When you have finished configuring the graph, save it to the project by selecting **File → Save** or by clicking the **Save** toolbar button (see below).

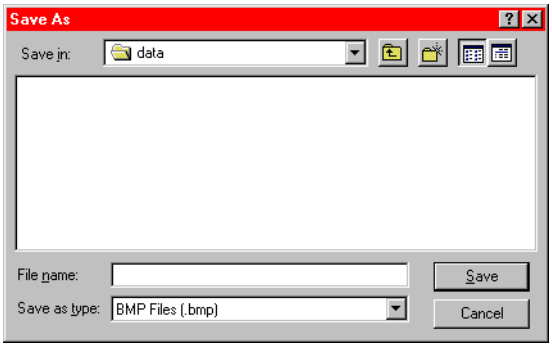


Saving a graph as a bitmap file

You can save a graph in bitmap (.bmp), JPEG (jpg), or TIFF (.tif) format for use elsewhere, such as in a report, as follows:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Save As.** The Save As window opens. See Figure 6-354.

Figure 6-354
Save As window



3. In the **File Name** edit box, type a file name.
4. In the **Save as type** combo box choose the file format for the saved graph from the following selections:
   - **BMP Files (*.bmp)**
   - **JPEG Files (*.jpg)**
   - **TIFF Files (*.tif)**
5. Indicate where the bitmap file is to be stored. The default file location is `C:\S4200\kiuser\projects\<ActiveProjectName>\tests\data\`. To store the bitmap file elsewhere, do one of the following in the Save As window:
   - **Option I**: In the **File Name** edit box, add the complete directory path and the file extension to the file name (for example, change **FileName** to **X:\OtherDirectory1\OtherDirectory2\FileName.bmp**, where **X** is a drive letter).
   - **Option II**: Browse for the correct project directory directly in the **Save In** combo box or use the next-file-level-up button, illustrated below.
6. Click the **Save** button.

Resetting certain graph properties to KITE defaults

CAUTION **You cannot undo the reset action.**

Using the **Reset** menu selection results in the following:

- Colors are restored to the defaults. This action applies to the text, axes, cursors, plots (series), and graph area (background and foreground).
- The graph size is restored to the default.
- The graph position is restored to the default.

NOTE *Zooms (the results of **Zoom In** and **Zoom Out**) are not affected. Zooms are not graph properties.*

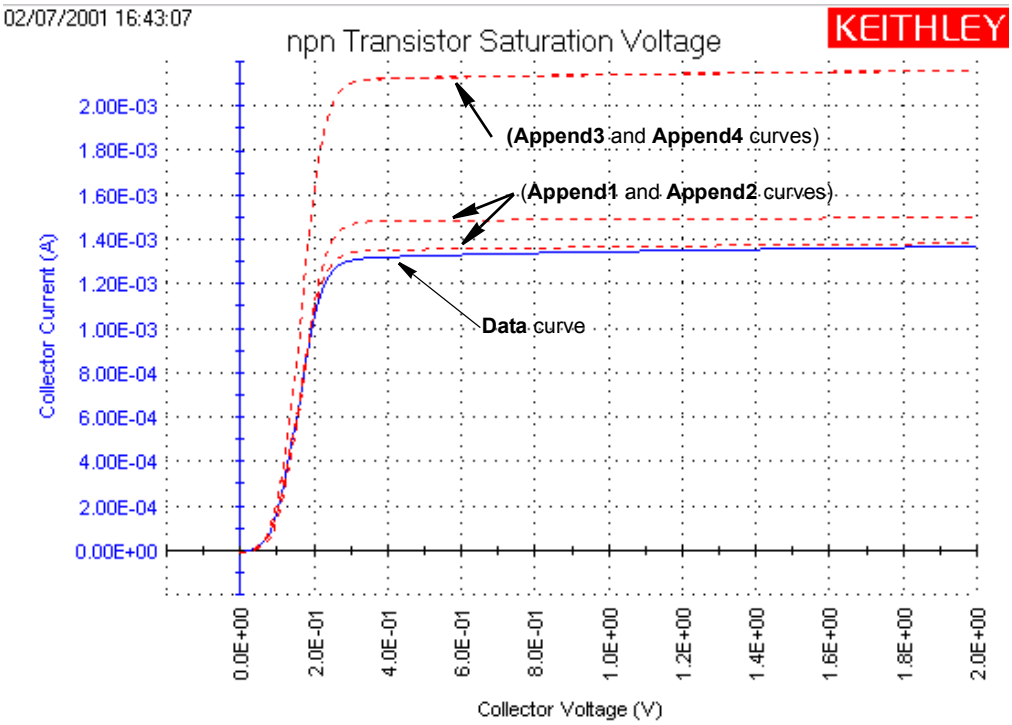
To initiate these changes, do the following:

1. In the **Graph** tab, display the **Graph Settings** menu by right-clicking on the graph or by selecting **Tools > Graph Settings**.
2. In the **Graph Settings** menu, select **Reset.** The graph resets as noted above (bulleted list).

Appending curves from multiple runs on a single graph

The **Append** execution feature appends (layers) curves from multiple runs on a single graph. In the graph of a specific test instance, the results of one or more **Append** type executions all append to the results of the last **Run** type execution. Figure 6-355 shows the **vcsat** curves for one **Run** type execution and four **Append** type executions on individual 2N3904 BJTs. Two of the **Append** curves (dashed lines) essentially overlap (the topmost curves appear almost as one curve).

Figure 6-355
Append-mode graph example



Append executions apply to an entire test sequence (a Device Plan or Subsite Plan) as well as to a solitary test. Each time the sequence is run, the results of each test in the sequence append to the appropriate graph (for more information about **Append** executions, refer to Append execution of tests, test sequences, and Project Plans).

Append worksheets

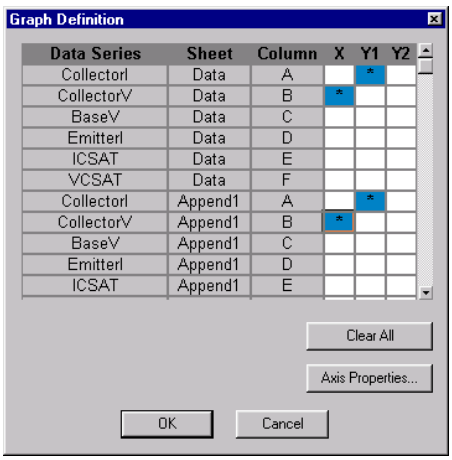
The data generated for each **Append** execution of a test is located in additional **Append** worksheets, where **n** designates the **n**th **Append** execution. Refer to Understanding and using Append worksheets of a Sheet tab

Append selections in the Graph Definition window

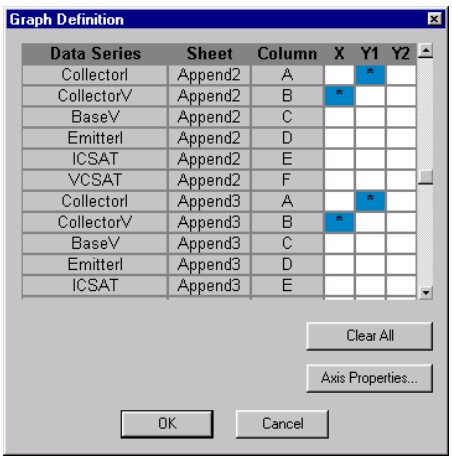
The Graph Definition window includes the added **Append** data. See Figure 6-356.

Figure 6-356
Append-curve definitions in Graph Definition window

As Initially displayed



After scrolling



In the Graph Definition window, you can modify the Append selections as follows:

- You can do the following globally, for data plotted on the **Graph** tab from the **Data** worksheet and *all* **Append** worksheets:
  - Globally add or change a Y axis plot parameter(s), by adding or changing the **Data** Y axis plot parameter(s).
  - Globally remove a Y axis plot parameter(s), by removing the **Data** Y axis plot parameter(s)
  - Globally change the X axis plot parameter, by changing the **Data** X axis plot parameter.
- You can do the following for an individual **Append**:
  - Add or change a Y axis plot parameter.
  - Remove a Y axis plot parameter.

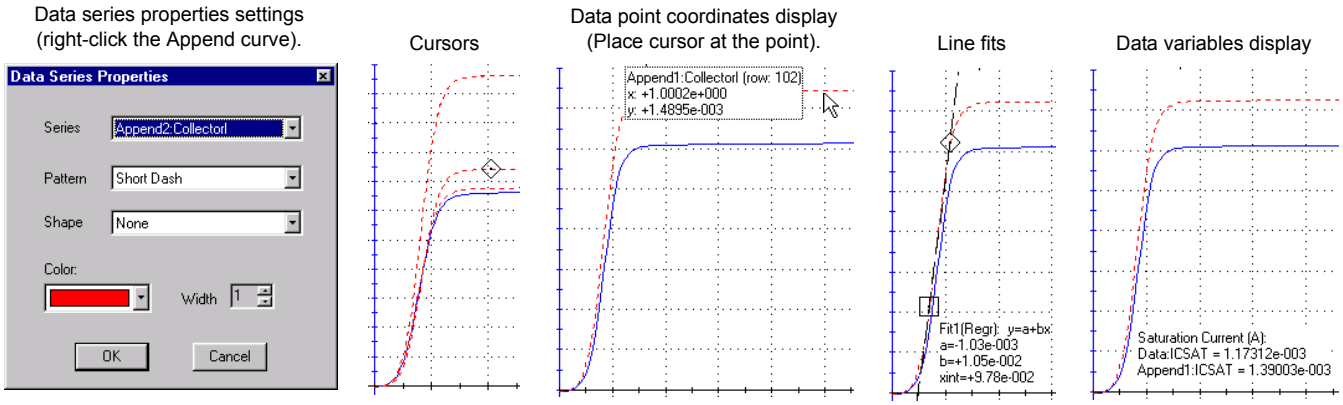
NOTE *You cannot individually select the X-axis plot parameter for an **Append** worksheet.*

Append curves

Working with Append curves

You can work with curves plotted from the **Append** worksheets in the same way as curves plotted from the **Data** worksheet. See Figure 6-357.

Figure 6-357
Functions that work with both Append and Run curves



For more information about these functions, refer to the following sections:

- **Data series properties**: Defining the plot properties of the graph: colors, line patterns, symbols, line widths.
- **Cursors**: Numerically displaying plot coordinates using cursors.
- **Coordinates**: Viewing plot coordinates and data series properties using the pointing device (mouse).
- **Line fits:** Performing on-graph line fits.
- **Data variables**: Numerically displaying extracted parameters and other data variables.

Permanently deleting Append curves

To permanently delete **Append** curves from the graph, delete the corresponding **Append** worksheets. To delete **Append** worksheets, refer to Deleting Append worksheets later in this section.

Analyzing test data using the Formulator

This subsection describes the following:

- Understanding the Formulator
- Starting the Formulator
- Becoming familiar with the Formulator window
- Understanding the Formulator functions
- Identifying data analysis requirements
- Creating an analysis formula
- Adding an analysis formula to the ITM or UTM
- Viewing analysis results in the Sheet tab Data worksheet
- Viewing analysis results in the Graph tab
- Editing formulas and constants

Understanding the Formulator

The **Formulator**, accessible from each ITM or UTM **Definition** tab, allows you to perform simple and complex data calculations on test data as well as on the results of other **Formulator** calculations. The **Formulator** provides a variety of computational functions, common mathematical operators, and common constants.

A formula created by the **Formulator** is an equation composed from a series of functions, operators, constants, and arguments. The next two subsections summarize how the **Formulator** applies these elements.

Formulator arguments and constants

A formula created using the **Formulator** performs calculations on any combination of the following:

- ITM or UTM test data.
- Secondary data created by other **Formulator** formulas.
- Standard constants in the list of constants.

Some of the functions operate on **Sheet** tab columns of values (vectors) only. Others operate on single values (scalars) only. Still others operate on both single values (scalars) and columns of values (vectors).

Likewise, the results of some calculations may be a column of values (vector) in the **Sheet** tab **Data** worksheet or a column containing only a single value (scalar).

Formulator functions and operators

The **Formulator** provides a variety of functions and operators. Some of these may be used for real-time, in-test calculations for ITM data (though not for UTM data). Others may be used only for post-test data computations. The next two subsections explain the differences.

Real-time functions, operators, and formulas

A formula containing *exclusively real-time* operators and functions is a real-time formula. If a real-time formula is specified as part of an ITM definition, it executes for each data point generated by the ITM, just after it is generated. The results of a real-time formula may be viewed in the **Sheet** tab **Data** worksheet or plotted during the test in the same way as test data.

NOTE *Real-time calculations do not apply to UTM data. A UTM **Data** worksheet does not update until the test is finished.*

The following operators and functions are real-time operators and functions:

- Operators: **`+, -, *, /, ^`** (exponentiation).
- Functions: **`ABS, DELTA, DIFF, EXP, INTEG, LN, LOG, SQRT`**
  (For function definitions, refer to Functions area later in this section).

The formula below is a real-time formula:

- `RESULT1 = ABS(DELTA(GATECURRENT))`

Real-time formulas execute as follows:

- If a real-time formula is created before the ITM has been run, the formula executes automatically during each run.
- If a real-time formula is created after an ITM has been run, the formula executes initially upon adding it to the ITM and automatically during each subsequent run.

Post-test-only functions and formulas

A formula containing *any one* (or more) of the remaining **Formulator** functions is a post test only formula. It executes only at the end of each run of the ITM or UTM in which the formula is defined. The results of a post test only formula may be viewed in the **Sheet** tab **Data** worksheet or plotted only at the end of a test.

The following functions are post test only functions:

AT, AVG, COND, EXPFIT, EXPFITA, EXPFITB, FINDD, FINDLIN, FINDU,FIRSTPOS, LASTPOS, LINFIT, LINFITSLP, LINFITXINT, LINFITYINT, LOGFIT, LOGFITA, LOGFITB, MAVG, MAX, MAXPOS, MIN, MINPOS, REGFIT, REGFITSLP, REGFITXINT, REGFITYINT, SUBARRAY, SUMMV, TANFIT, TANFITSLP, TANFITXINT, TANFITYINT

The formula below is a post test only formula, because MAVG is a post test only function:

- `RESULT2 = MAVG(ABS(DELTA(GATECURRENT)),3)`

Post test only formulas execute as follows:

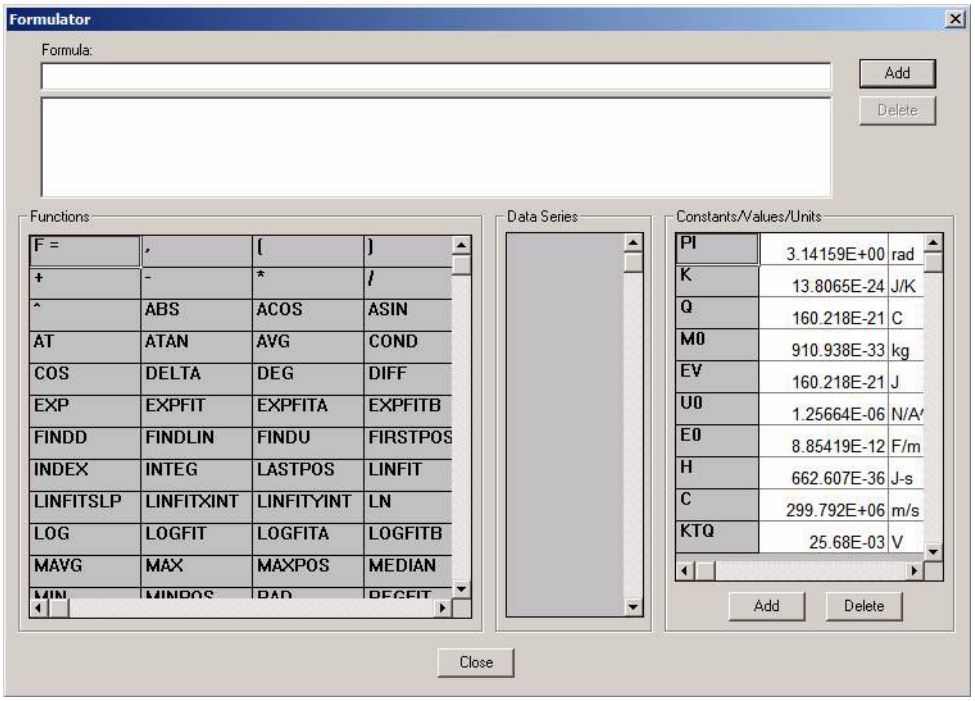
- If a post test only formula is created before the ITM or UTM has been run, the formula executes automatically at the conclusion of each run.
- If post test only formula is created after an ITM or UTM has been run, the formula executes initially upon adding it to the UTM or ITM and automatically at the conclusion of each subsequent run.

Starting the Formulator

Start the **Formulator** as follows:

1. Open the **Definition** tab for the test data that you wish to analyze by clicking on the appropriate ITM or UTM in the Project Navigator. The **Definition** tab appears.
2. In the **Definition** tab, click on the **Formulator** button. The Formulator window opens. See Figure 6-358.

Figure 6-358
Formulator window, unconfigured



Becoming familiar with the Formulator window

This subsection summarizes the significance/use of each **Formulator** window feature.

Formula area

The Formula boxes

The two boxes in the **Formula** area of the window are used as follows:

- The upper **Formula** box is used to create new formulas or edit existing formulas.
- The lower **Formula** box displays the formulas that have been created for the ITM or UTM.

The Add formulas button

A click of the **Add** button does the following:

- Starts calculation execution for the formula in the upper **Formula** box.
- Moves a formula from the upper **Formula** box into the collection of formulas displayed in the lower **Formula** box.

The Delete formulas button

The **Delete** button deletes a formula that is selected in the lower box.

Functions area

The functions area displays a matrix of mathematical operators and functions, as well as a generic `F=` that can be used in place of a variable name to complete the equation. When you click a button, the function is added to the equation in the upper window, at the cursor position.

NOTE *When you **Add** an equation containing **F=**, KITE adds a numeric suffix to the **F**, for example, **F1**, **F2**, **F3**.*

For details about the available functions, refer to the subsequent subsection Understanding the Formulator functions.

Columns area

Lists the names of all columns in the **Data** worksheet of the **Sheet** tab (note that columns created by some functions may contain only a single value).

Constants/Values/Units area

Provides constants that may be conveniently inserted by name.

Constants list

To ensure clarity, the default symbols in the constants list are identified below:

- **PI** π
- **K** Boltzmann constant
- **Q** Charge on an electron
- **M0** Electron mass
- **EV** Electron volt
- **U0** Permeability
- **E0** Permittivity of a vacuum
- **H** Planck constant

- **C** Speed of light
- **KTQ** Thermal voltage

When you click the button next to a constant in the constants list, the constant is added to the equation in the upper window, at the cursor position.

NOTE *Both the values and units of constants in the constants list can be edited directly.*

Add constants button

Clicking the **Add** button opens a window that allows you to add a new constant to the constants list.

Delete constants button

Clicking the **Delete** button opens a window that allows you to delete a constant from the constants list.

Understanding the Formulator functions

NOTE *In the Formulator function descriptions, 10 point Courier distinguishes function format parameter names from other text.*

The **Formulator** functions, as well as operators and constants, can be used singly or in various combinations to create simple or complex analysis equations.

Multiple functions can be nested. For example, in one equation you can calculate the following:

- Calculate a series of moving averages for a column of data (vector) in the **Data** worksheet, using the **MAVG** function.
- Find the maximum value of the **MAVG** averages, using the **MAX** function.
- Multiply the **MAX** found value by a constant.

The equation below illustrates this use of nested **Formulator** functions.

```
MAXDIFF = 10*MAX(MAVG(ColumnA))
```

The degree (number of levels) of nesting is unlimited.

The purpose, format, and arguments for the above functions and all other functions available in the **formulator** are described below. An example is given in each case.

NOTE *To be consistent with the format of an Excel spreadsheet, row 1 of a **Sheet** tab **Data** worksheet contains column headings. Therefore, when the row number (index) of a column (vector) is specified as a function argument, do not insert 1.*

*Keithley Instruments recommends using the function **FIRSTPOS** :*

[format: `FIRSTPOS(DataWorksheetColumn)`] as the argument for the first value in a vector.

*The similar function **LASTPOS**:*

[format: `LASTPOS(DataWorksheetColumn)`] may be used for the last value in the vector.

*In **Graph** tab graphs, you can directly perform composite line fits that are equivalent to the following groups of individual Formulator line fits: **EXPFIT**, **EXPFITA**, and **EXPFITB**; **LINFIT**, **LINFITSLP**, **LINFITXINT**, and **LINFITYINT**; **LOGFIT**, **LOGFITA**, and **LOGFITB**; **REGFIT**, **REGFITSLP**, **REGFITXINT**, and **REGFITYINT**; **TANFIT**, **TANFITSLP**, **TANFITXINT**, and **TANFITYINT**. However, the fit lines and parameters only display in the graphs. They are unavailable for use in calculations. Refer to Table 6-14 and to Performing on-graph line fits.*

Table 6-14
Correspondence between Graph tab and Formulator line fits

Formulator fit functions*				**Corresponding Graph tab line fit**
LINFIT,	LINFITYINT	LINFITSLP	LINFITXINT	Linear
REGFIT	REGFITYINT	REGFITSLP	REGFITXINT	Regression
EXPFIT	EXPFITA	EXPFITB	————	Exponential
LOGFIT	LOGFITA	LOGFITB	————	Logarithmic
TANFIT	TANFITYINT	TANFITSLP	TANFITXINT	Tangent

* These functions calculate individual fit lines and parameters that may be used in other calculations. By contrast, the Graph tab calculates and displays **only** the fit line and all fit parameters.

Formulator function reference

Each of the Model 4200-SCS Formulator functions are described below.

ABS: Formulator function

Purpose Calculates the absolute value of each value in the designated column (vector) or the absolute value of any operand.

Format
```
ABS(X)
```
Where: `X =` The name of any column (vector) listed under **Columns** or any operand.

Example
```
F2 = ABS(GateI)
```

Remarks This function can be used to perform calculations in real time, while a test is executing.

ACOS: Formulator function

Purpose Returns the arc cosine of each value in a designated column (vector) under **Columns** or any operand.

Format
```
ACOS(X)
```
```
Where: X = The name of any column (vector) listed under Columns
or any operand.
```

Example
```
F1 = ACOS(DRAINI)
```

Remarks Returns the value in radians.

ASIN: Formulator function

Purpose Returns the arc sine of each value in a designated column (vector) under **Columns** or any operand.

Format
```
ASIN(X)
```
Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = ASIN(DRAINI)
```

Remarks Returns the value in radians.

AT: Formulator function

Purpose Extracts and returns a single value from a column (vector).

Format
```
AT(V, POS)
```
Where: *V* = The name of any column (vector) listed under **Columns**.
POS = The row number of column V where the single value is located.

Example
```
IDSAT = AT(DRAINI, 36)
```

ATAN: Formulator function

Purpose Returns the arc tangent of each value in a designated column (vector) under **Columns** orany operand.

Format
```
ATAN(X)
```
Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = ATAN(DRAINI)
```

Remarks Returns the value in radians.

AVG: Formulator function

Purpose Returns the average of all values in the column (vector).

Format
```
AVG(V)
```
Where: V = The name of any column (vector) listed under **Columns**.

Example
```
LEAKAGE = AVG(GATEI)
```

Remarks Refer also to the **MAVG** function.

COND: Formulator function

Purpose Returns one of two user-defined expressions (EXP3 or EXP4), depending on the comparison of two other user-defined expressions (EXP1 and EXP2).

- If EXP1 < EXP2, then EXP3 is returned.
- If EXP1 ≥ EXP2, then EXP4 is returned.

Format
```
COND(EXP1, EXP2, EXP3, EXP4)
```
Where: EXP1, EXP2, EXP3, and EXP4 = mathematical expressions created using valid **Formulator** functions, operators, and operands.

Example
```
CLIPCURRENT = COND(DRAINI, 1E-6, DRAINI, 1E-6)
```

COS: Formulator function

Purpose Returns the cosine of each value operand.

Format
```
COS(X)
```
Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = COS(DRAINI)
```

Remarks Returns the value in radians.

DEG: Formulator function

Purpose The DEG function converts an angle value in Radians to Degrees.

Format
```
DEG(X)
```
Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = DEG(ANGLE)
```

Remarks Returns the value in degrees.

DELTA: Formulator function

Purpose Returns the differences between the adjacent values in a column (vector). That is, for column V, DELTA returns (V2 – V1),(V3 – V2), and so on.

Format
```
DELTA(V)
```
Where: V = The name of any column (vector) listed under **Columns**.

Example
```
GM = DELTA(DRAINI)/DELTA(GATEV)
```

Remarks This function can be used to perform calculations in real time, while a test is executing.

DIFF: Formulator function

Purpose For all of the values in two selected columns (vectors), returns a third column (vector) containing the difference coefficients. Each coefficient is calculated as follows:

$$\frac{\text{The difference between a pair of adjacent values in the first column, } V1}{\text{The difference between the corresponding values in the second column, } V2}$$

That is, for columns V1 and V2, DIFF returns the following: $(V1_2 - V1_1)/(V2_2 - V2_1)$, $(V1_3 - V1_2)/(V2_3 - V2_2)$, and so on.

Format
```
DIFF(V1, V2)
```
Where: `V1` = The name of any column (vector) listed under **Columns**.
`V2` = The name of any column (vector) listed under **Columns**.

Example
```
GM = DIFF(DRAINI, GATEV)
```

Remarks This function can be used to perform calculations in real time, while a test is executing.

EXP: Formulator function

Purpose Returns the exponential, e^{value}, for each value in a column (vector) or for any operand.

Format
```
EXP(X)
```
Where: `X` = The name of any column (vector) listed under **Columns** or any operand.

Example
```
NEWCURRENT = CURRENT*EXP(ANODEV)
```

Remarks This function can be used to perform calculations in real time, while a test is executing.

EXPFIT: Formulator function

Purpose Performs an exponential fit as follows:

- Fits the following exponential relationship to a specified *range* of values in two columns (vectors): one column, `VX`, containing X values and the other column, `VY`, containing Y values:
  $Y = EXPFITA * e^{(EXPFITB * X)}$
- where EXPFITA and EXPFITB are fit constants.
- Using the above exponential relationship, returns a new column (vector) containing Y values calculated from *all* X values in column `VX`.

Format
```
EXPFIT(VX, VY, STARTPOS, ENDPOS)
```
Where: `VX` = The name of any column (vector) listed under **Columns**.
`VY` = The name of any column (vector) listed under **Columns**.
`STARTPOS` = For the range of X and Y values to be exponentially fitted, the row number (index) of the starting values.
`ENDPOS` = For the range of X and Y values to be exponentially fitted, the row number (index) of the ending values.

Example
```
DIODEI = EXPFIT(ANODEV, ANODEI, 2, LASTPOS(ANODEV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

EXPFITA: Formulator function

Purpose Performs an exponential fit as follows:

- Fits the following exponential relationship to a specified range of values in two columns (vectors)—one column, VX, containing X values and the other column, VY, containing Y values:
  $Y = EXPFITA * e^{(EXPFITB * X)}$
- where EXPFITA and EXPFITB are fit constants.
- Returns the value of the constant EXPFITA in the relationship above.

Format

```
EXPFITA(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be exponentially fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be exponentially fitted, the row number (index) of the ending values.

Example

```
DIODEOFFSET = EXPFITA(ANODEV, ANODEI, 2, LASTPOS(ANODEV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

EXPFITB: Formulator function

Purpose Performs an exponential fit as follows:

- Fits the following exponential relationship to a specified range of values in two columns (vectors)—one column, VX, containing X values and the other column, VY, containing Y values:
  $Y = EXPFITA * e^{(EXPFITB * X)}$
- where EXPFITA and EXPFITB are fit constants.
- Returns the value of the constant EXPFITB in the relationship above.

Format

```
EXPFITB(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be exponentially fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be exponentially fitted, the row number (index) of the ending values.

Example

```
DIODEIDEALITY = 1/(EXPFITB(ANODEV, ANODEI, 2,
LASTPOS(ANODEV))*0.0257)
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

FINDD: Formulator function

Purpose (Find down) Given a column (vector) V, beginning at START, FINDD searches *down* the column until it finds a value that matches the user-specified value X. Then it returns the row number (index) of that value. If FINDD does not find an exact match for X, it returns the row number (index) of the V value that is closest to X.

Format

```
FINDD(V, X, START)
```

Where: V = The name of any column (vector) listed under **Columns**.
X = Any value (which may be the result of another calculation[s]).
START = The row number (index) of the starting value for the search.

Example

```
IF = AT(ANODEI, FINDD(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Remarks Refer also to the similar functions FINDLIN (Find using linear interpolation) and FINDU (Find up).

FINDLIN: Formulator function

Purpose (Find using linear interpolation) Given a column (vector) V, beginning at START, FINDLIN searches down the column until it finds a value that is closest (but does not exceed) the user-specified value X. Linear interpolation is then used to determine its decimal location between the found value and the next value in the column (vector). The returned index number (in decimal format) indicates the position of the specified value.

Purpose For example, assume you want to use FINDLIN to locate value 6 in the following array.

(Index 1)V

(Index 2)1

(Index 3)4

(Index 4)8

The search finds the index marker that is closest to (but does not exceed) 6. In this case, Index 3 is the closest. Linear interpolation is then used to determine the decimal position of the specified value (6) that is between Index 3 (value 4) and Index 4 (value 8). Value 6 is halfway between Index 3 and Index 4. Therefore, FINDLIN will return Index 3.5.

Format

```
FINDD(V, X, START)
```

Where: V = The name of any column (vector) listed under **Columns**.
X = Any value (which may be the result of another calculation[s]).
START = The row number (index) of the starting value for the search.

Example

```
IF = AT(ANODEI, FINDLIN(ANODEV, 0.7, FIRSTPOS(ANODEV)))
```

Remarks Refer also to the similar functions FINDD (Find down) and FINDU (Find up).

FINDU: Formulator function

Purpose (Find up) Given a column (vector) V, beginning at START, FINDU searches *up* the column until it finds a value that matches the user-specified value X. It then returns the row number (index) of that value. If FINDU does not find an exact match for X, it returns the row number (index) of the V value that is closest to X.

Format
```
FINDU(V, X, STARTPOS)
```
Where: V = The name of any column (vector) listed under **Columns**.
X = Any value (which may be the result of another calculation[s]).
STARTPOS = The row number (index) of the starting value for the search.

Example
```
IF = AT(ANODEI, FINDU(ANODEV, 0.7, LASTPOS(ANODEV)))
```

Remarks Refer also to the similar functions FINDLIN (Find using linear interpolation) and FINDD (Find down).

FIRSTPOS: Formulator function

Purpose Returns the row number (index) of the first value in a column (vector), typically the number 2.

Format
```
FIRSTPOS(V)
```
Where: V = The name of any column (vector) listed under **Columns.**

Example
```
STARTOFARRAY = FIRSTPOS(DRAINI)
```

Remarks Refer also to the function LASTPOS.

INDEX: Formulator Function

Purpose Will return a specified amount of points starting with a specified value and consecutive values incremented by one.

Format
```
INDEX(START, N)
```
```
Where:                START = The starting value.
                      N = The number of data points to be
included.
```

Example
```
INDEX20 = INDEX(5, 20)
```

Remarks The example will produce a new column labeled INDEX20 containing 20 values starting with the value of 5 and ending with a value of 24.

INTEG: Formulator function

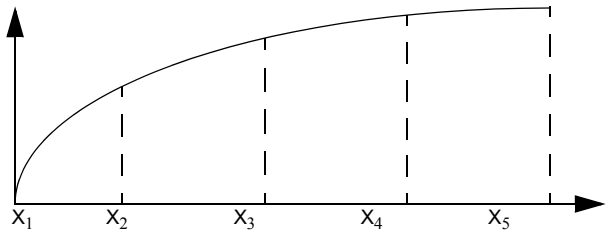
Purpose From two columns (vectors) VX and VY, each one containing N values, the INTEG function returns a third column (vector) containing a series of numerical integrals A_n, where n = 1, 2,..., N-1, N. Each integral approximates the area under the parametric

curve created by plotting the first n values in VY against the first n values in VX. For n = 1, $A_n = 0$. For all other values of n, each integral A_n corresponds to the following relationship:

$$A_n = \sum_{i=1}^{i=(n-1)} (X_{i+1} - X_i) \cdot (Y_{i+1} + Y_i)/2$$

For example, for the curve below,

Figure 6-359
INTEG: Formulator function



INTEG returns a column (vector) containing A_1 equal to 0 (zero area at the start of a curve, at X_1) and A_2, A_3, A_4 and A_5 equal to curve areas as follows:

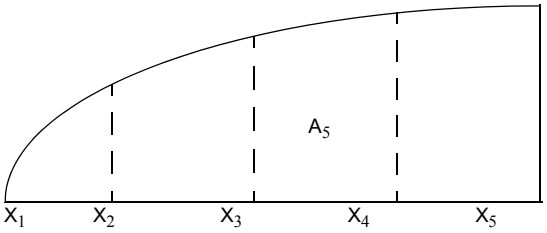
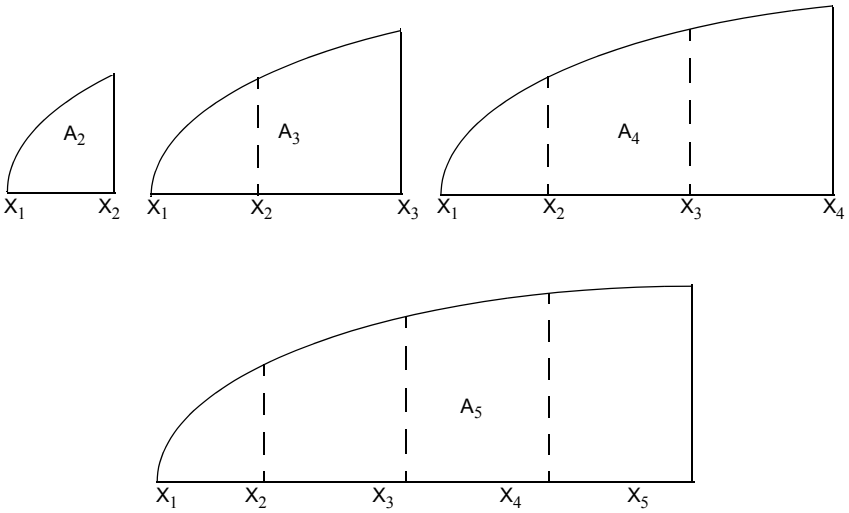


Figure 6-360
INTEG: Formulator function



Format INTEG(VX, VY)

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.

Example `QBD = INTEG(TIME, GATEI)`

Remarks This function can be used to perform calculations in real time, while a test is executing.

LASTPOS: Formulator function

Purpose Returns the row number (index) of the last value in a column (vector).

Format `LASTPOS(V)`

Where: V = The name of any column (vector) listed under **Columns**.

Example `NUMSWEEPPTS = LASTPOS(COLLECTORI)`

Remarks Refer also to the function `FIRSTPOS`.

LINFIT: Formulator function

Purpose Performs the following:

- Finds a linear equation of the form Y = a + bX from two sets of X and Y values selected from two columns (vectors), VX and VY. This equation corresponds to a line drawn through two points on a curve that is created by plotting the values in VY against the values in VX. The two points are specified by the arguments STARTPOS and ENDPOS.
- Using the linear equation, returns a new column (vector) containing Y values calculated from all X values in column VX.

Format `LINFIT(VX, VY, STARTPOS, ENDPOS)`

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = The row number (index) of the first set of X and Y values.
ENDPOS = The row number (index) of the second set of X and Y values.

Example `RESISTORFIT = LINFIT (rESV, RESI, FIRSTPOS(rESV), LASTPOS(rESV))`

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

To return a linear regression fit for two columns (vectors), use the REGFIT function.

LINFITSLP: Formulator function

Purpose Performs the following:

- Finds a linear equation of the form Y = a + bX from two sets of X and Y values selected from two columns (vectors), VX and VY. This equation corresponds to a line drawn through two points on a curve that is created by plotting the values in VY

against the values in VX. The two points are specified by the arguments STARTPOS and ENDPOS.
- Returns the slope of the linear equation (value of b in Y = a + bX).

Format

```
LINFITSLP(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = The row number (index) of the first set of X and Y values.
ENDPOS = The row number (index) of the second set of X and Y values.

Example

```
RESISTANCE = 1/LINFITSLP(rESV, RESI, FIRSTPOS(rESV),
LASTPOS(rESV))
```

Remarks

If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

To return the slope of a linear *regression* fit for two columns (vectors), use the REGFITSLP function.

LINFITXINT: Formulator function

Purpose

Performs the following:

- Finds a linear equation of the form Y = a + bX from two sets of X and Y values selected from two columns (vectors), VX and VY. This equation corresponds to a line drawn through two points on a curve that is created by plotting the values in VY against the values in VX. The two points are specified by the arguments STARTPOS and ENDPOS.
- Returns the X intercept of the linear equation (value of -a/b in Y = a + bX).

Format

```
LINFITXINT(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = The row number (index) of the first set of X and Y values.
ENDPOS = The row number (index) of the second set of X and Y values.

Example

```
EARLYV = LINFITXINT(COLLECTORV, COLLECTORI, 56, 75)
```

Remarks

If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

To return the X intercept of a linear regression fit for two columns (vectors), use the REGFITXINT function.

LINFITYINT: Formulator function

Purpose

Performs the following:

- Finds a linear equation of the form Y = a + bX from two sets of X and Y values selected from two columns (vectors), VX and VY. This equation corresponds to a line drawn through two points on a curve that is created by plotting the values in VY

against the values in VX. The two points are specified by the arguments STARTPOS and ENDPOS.

- Returns the Y intercept of the linear equation (value of a in Y = a + bX).

Format

```
LINFITYINT(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = The row number (index) of the first set of X and Y values.
ENDPOS = The row number (index) of the second set of X and Y values.

Example

```
OFFSET = LINFITYINT(GATEV, GATEI, FIRSTPOS(GATEV),
LASTPOS(GATEV))
```

Remarks

If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

To return the Y intercept of a linear regression fit for two columns (vectors), use the REGFITYINT function.

LN: Formulator function

Purpose

Returns the base-e (natural, Napierian) log of each value in a designated column (vector) or the Napierian log of any operand.

Format

```
LN(X)
```

Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example

```
DIODEV = LN(ANODEI)*0.026
```

Remarks

This function can be used to perform calculations in real time, while a test is executing.

LOG: Formulator function

Purpose

Returns the base-10 log of each value in a designated column (vector) or the base-10 log of any operand.

Format

```
LOG(X)
```

Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example

```
F1 = LOG(DRAINI)
```

Remarks

This function can be used to perform calculations in real time, while a test is executing.

LOGFIT: Formulator function

Purpose

Performs a base-10 log-linear fit as follows:

- Fits the following logarithmic relationship to a specified *range* of values in two columns (vectors) (one column, VX, containing X values and the other column, VY,

containing Y values):
Y = LOGFITA + LOGFITB* log (X)

- where LOGFITA and LOGFITB are fit constants.
- Using the above logarithmic relationship, returns a new column (vector) containing Y values calculated from *all* X values in column VX.

Format `LOGFIT(VX, VY, STARTPOS, ENDPOS)`

Where: `VX` = The name of any column (vector) listed under **Columns**.
`VY` = The name of any column (vector) listed under **Columns**.
`STARTPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the starting values.
`ENDPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the ending values.

Example
```
GOODFIT = LOGFIT(GATEV, DRAINI, 30, 50)
```

Remarks If a `VX` or `VY` value at either `STARTPOS` or `ENDPOS` is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

LOGFITA: Formulator function

Purpose Performs a base-10 log-linear fit as follows:

- Fits the following logarithmic relationship to a specified *range* of values in two columns (vectors) (one column, `VX`, containing X values and the other column, `VY`, containing Y values):
Y = LOGFITA + LOGFITB* log (X)
- where LOGFITA and LOGFITB are fit constants.
- Returns the value of the constant LOGFITA in the relationship above.

Format `LOGFITA(VX, VY, STARTPOS, ENDPOS)`

Where: `VX` = The name of any column (vector) listed under **Columns**.
`VY` = The name of any column (vector) listed under **Columns**.
`STARTPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the starting values.
`ENDPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the ending values.

Example
```
OFFSET = LOGFITA(GATEV, DRAINI, 30, 50)
```

Remarks If a `VX` or `VY` value at either `STARTPOS` or `ENDPOS` is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

LOGFITB: Formulator function

Purpose Performs a base-10 log-linear fit as follows:

- Fits the following logarithmic relationship to a specified *range* of values in two columns (vectors) (one column, `VX`, containing X values and the other column, `VY`, containing Y values):
Y = LOGFITA + LOGFITB* log (X)

- where LOGFITA and LOGFITB are fit constants.
- Returns the value of the constant LOGFITB in the relationship above.

Format `LOGFITB(VX, VY, STARTPOS, ENDPOS)`

Where: `VX` = The name of any column (vector) listed under **Columns**.
`VY` = The name of any column (vector) listed under **Columns**.
`STARTPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the starting values.
`ENDPOS` = For the range of X and Y values to be logarithmically fitted, the row number (index) of the ending values.

Example
```
FACTOR = LOGFITB(GATEV, DRAINI, 30, 50)
```

Remarks If a `VX` or `VY` value at either `STARTPOS` or `ENDPOS` is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

MAVG: Formulator function

Purpose Returns a new column (vector) consisting of the moving averages of successive groups of data points from another column (vector). The number of data points in a group is user-configurable.

Format `MAVG(V, N)`

Where: `V` = The name of any column (vector) listed under **Columns**.
`N` = The number of data points to be averaged in each group.

Example
```
FILTER = MAVG(GATEI, 3)
```

Remarks If `N` = 3 and `V` contains the 12 values $X_1, X_2, X_3, X_4, X_5, \ldots, X_{10}, X_{11}, X_{12}$, then `MAVG(V,N)` returns a column (vector) containing the following values:

#REF, (X1 + X2 + X3)/3, (X2 + X3+ X4)/3, (X3 + X4+ X5)/3, .. (X10 + X11+ X12)/3, #REF

The new column's values may contain instances of #REF (as shown above) because MAVG uses cells from both sides of the target cell for its calculation.

MAX: Formulator function

Purpose Searches all values in a column (vector) and returns the maximum value.

Format `MAX(V)`

Where: `V` = The name of any column (vector) listed under **Columns**.

Example
```
MAXGM = MAX(DIFF(DRAINI, GATEV))
```

MAXPOS: Formulator function

Purpose Searches all values in a column (vector), finds the maximum value, and returns the row number (index) of the maximum value.

Format `MAXPOS(V)`

Where: `V` = The name of any column (vector) listed under **Columns**.

Example
```
PEAKSTRESS = AT(GATEV, MAXPOS(SUBSTRATEI))
```

MEDIAN: Formulator function

Purpose Searches all values in a column (vector), finds the middle point of that column used, and returns the value.

Format `MEDIAN(V)`

`Where V = The name of any column (vector) listed under Columns.`

Example
```
MEDIANI = MEDIAN(DRAINI)
```

MIN: Formulator function

Purpose Searches all values in a column (vector) and returns the minimum value.

Format `MIN(V)`

Where: `V` = The name of any column (vector) listed under **Columns**.

Example
```
SMALLESTI = MIN(DRAINI)
```

MINPOS: Formulator function

Purpose Searches all values in a column (vector), finds the minimum value, and returns the row number (index) of the minimum value.

Format `MINPOS(V)`

Where: `V` = The name of any column (vector) listed under **Columns**.

Example
```
LOCATION = MINPOS(DRAINI)
```

RAD: Formulator function

Purpose The RAD function converts an angle value in Degrees to Radians.

Format `RAD(X)`

Where: `X` = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = RAD(ANGLE)
```

Remarks Returns the value in radians.

REGFIT: Formulator function

Purpose Performs a linear regression fit as follows:

- Fits the following relationship, of the form Y = a + bX, to a specified *range* of values in two columns (vectors) (column VX containing X values and column VY containing Y values):
  Y = REGFITYINT + REGFITSLP * X
  where REGFITSLP and REGFITYINT are slope and Y-intercept constants.
- Using the above linear relationship, returns a new column (vector) containing Y values calculated from *all* X values in column VX.

Format

```
REGFIT(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be fitted, the row number (index) of the ending values.

Example

```
COLLECTORFIT = REGFIT(COLLECTORV, COLLECTORI, 25,
LASTPOS(COLLECTORV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

REGFITSLP: Formulator function

Purpose Fits the following relationship, of the form Y = a + bX, to a specified *range* of values in two columns (vectors) (column VX containing X values and column VY containing Y values):
Y = REGFITYINT + REGFITSLP * X
where REGFITSLP and REGFITYINT are slope and Y-intercept constants.

- Returns the value of the slope constant REGFITSLP in the relationship above.

Format

```
REGFITSLP(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be fitted, the row number (index) of the ending values.

Example

```
COLLECTORRES = 1/REGFITSLP(COLLECTORV, COLLECTORI, 25,
LASTPOS(COLLECTORV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

REGFITXINT: Formulator function

Purpose Fits the following relationship, of the form Y = a + bX, to a specified *range* of values in two columns (vectors) (column VX containing X values and column VY containing Y values):
Y = REGFITYINT + REGFITYINT* X
where REGFITSLP and REGFITYINT are slope and Y-intercept constants.

- Returns the value of the X intercept for relationship above. (–REGFITYINT/REGFITSLP).

Format

```
REGFITXINT(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be fitted, the row number (index) of the ending values.

Example

```
EARLYV = REGFITXINT(COLLECTORV, COLLECTORI, 25,
LASTPOS(COLLECTORV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

REGFITYINT: Formulator function

Purpose Fits the following relationship, of the form Y = a + bX, to a specified *range* of values in two columns (vectors) (column VX containing X values and column VY containing Y values):
Y = REGFITYINT + REGFITSLP * X
where REGFITSLP and REGFITYINT are slope and Y-intercept constants.

- Returns the value of the Y intercept (rEGFITYINT) for relationship above.

Format

```
REGFITYINT(VX, VY, STARTPOS, ENDPOS)
```

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
STARTPOS = For the range of X and Y values to be fitted, the row number (index) of the starting values.
ENDPOS = For the range of X and Y values to be fitted, the row number (index) of the ending values.

Example

```
OFFSET = REGFITYINT(COLLECTORV, COLLECTORI, 25,
LASTPOS(COLLECTORV))
```

Remarks If a VX or VY value at either STARTPOS or ENDPOS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

SIN: Formulator function

Purpose Returns the sine of each value in a designated column (vector) under **Columns** or any operand.

Format `SIN(X)`

Where: X = The name of any column (vector) listed under Columns or any operand.

Example `F1 = SIN(DRAINI)`

Remarks Returns the value in radians.

SQRT: Formulator function

Purpose Returns the square root of each value in a designated column (vector) or the square root of any operand.

Format `SQRT(X)`

Where: `X` = The name of any column (vector) listed under **Columns** or any operand.

Example `TWO = SQRT(4)`

Remarks A negative value of `X` returns **#REF** in the **Data** worksheet.

This function can be used to perform calculations in real time, while a test is executing.

STDEV: Formulator function

Purpose Returns the standard deviation of all values in the column (vector).

Format `STDEV(V)`

Where: `V` = The name of any column (vector) listed under **Columns**.

Example `LEAKAGE = STDEV(GATEI)`

Remarks Returns the standard deviation.

SUBARRAY: Formulator function

Purpose Returns a new column (vector) containing a specified range of the values from an existing column (vector). For example, given an existing column (vector), $V_{exist,}$ containing values in rows 2 through 60, you could use `SUBARRY` to return a new column (vector), $V_{new,}$ containing only the values from rows 20 through 40 of V_{exist}.

Format `SUBARRAY(V, STARTPOS, ENDPOS)`

Where: `V` = The name of any column (vector) listed under **Columns**.
`STARTPOS` = The row number (index) of the existing value that you choose to become the first value in the new column (vector).

`ENDPOS` = The row number (index) of the existing value that you choose to become the last value in the new column (vector).

Example

```
SUB1 = SUBARRAY(rESV, 10, 20)
```

Remarks If `STARTPOS` and `ENDPOS` are invalid numbers, the function returns #REF as the result.

SUMMV: Formulator function

Purpose Returns a column (vector) VY that consists of moving summation of a column (vector) V. The n^{th} value in VY (Y_n) is the sum of the n^{th} and preceding values in V. This relationship may be expressed mathematically as follows:

$$Y_n = \sum_{i=1}^{i=n} X_i$$ where X_i are the values in column (vector) V.

The following example illustrates the SUMMV function numerically:

Table 6-15
SUMMV: Formulator function

V	VY = SUMMV(V)
1.0000	1.0000
2.0000	3.0000
3.0000	6.0000
4.0000	10.0000
•	•
•	•
•	•

Format
```
SUMMV(V)
```
Where: V = The name of any column (vector) listed under **Columns**.

Example
```
F1 = SUMMV(BASEI)
```

Example
```
PSISPSIO = SUMMV((1-CQADJ/COX)*DELTA(VGS))*DOPETYPE
```

TAN: Formulator Function

Purpose Returns the tangent of each value in a designated column (vector) under **Columns** or any operand.

Format
```
TAN(X)
```
Where: X = The name of any column (vector) listed under **Columns** or any operand.

Example
```
F1 = TAN(DRAINI)
```

Remarks Returns the value in radians.

TANFIT: Formulator function

Purpose Performs the following:

- Finds a linear equation of the form Y = a + bX from two columns (vectors), VX and VY. This equation corresponds to a tangent of the curve that is created by plotting the values in VY against the values in VX. The value at which the tangent is found is specified by the argument POS.

- Using the linear equation, returns a new column (vector) containing Y values calculated from all X values in column VX.

Format TANFIT(VX, VY, POS)

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
POS = The row number (index) of the value where the tangent is to be found.

Example
```
VTFIT = TANFIT(GATEV, DRAINI, MAXPOS(GM))
```

Remarks If a VX or VY value at POS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

TANFITSLP: Formulator function

Purpose Finds a linear equation of the form Y = a + bX from two columns (vectors), VX and VY. This equation corresponds to a tangent of the curve that is created by plotting the values in VY against the values in VX. The value at which the tangent is found is specified by the argument POS.

- Returns the slope of the linear equation (value of b in Y = a + bX).

Format TANFITSLP(VX, VY, POS)

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
POS = The row number (index) of the value where the tangent is to be found.

Example
```
VTSLOPE = TANFITSLP(GATEV, DRAINI, MAXPOS(GM))
```

Remarks If a VX or VY value at POS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

TANFITXINT: Formulator function

Purpose Finds a linear equation of the form Y = a + bX from two columns (vectors), VX and VY. This equation corresponds to a tangent of the curve that is created by plotting the values in VY against the values in VX. The value at which the tangent is found is specified by the argument POS.

- Returns the X intercept of the linear equation (value of -a/b in Y = a + bX).

Format TANFITXINT(VX, VY, POS)

Where: VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
POS = The row number (index) of the value where the tangent is to be found.

Example
```
VT = TANFITXINT(GATEV, DRAINI, MAXPOS(GM))
```

Remarks If a VX or VY value at POS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

TANFITYINT: Formulator function

Purpose Finds a linear equation of the form Y = a + bX from two columns (vectors), VX and VY. This equation corresponds to a tangent of the curve that is created by plotting the values in VY against the values in VX. The value at which the tangent is found is specified by the argument POS.

- Returns the Y intercept of the linear equation (value of a in Y = a + bX).

Format

```
TANFITYINT(VX, VY, POS)
```

Where:
VX = The name of any column (vector) listed under **Columns**.
VY = The name of any column (vector) listed under **Columns**.
POS = The row number (index) of the value where the tangent is to be found.

Example

```
OFFSET = TANFITYINT(GATEV, DRAINI, GMMAX)
```

Remarks If a VX or VY value at POS is an invalid number (that is, the value is **#REF**), the function will not return a valid result.

Identifying data analysis requirements

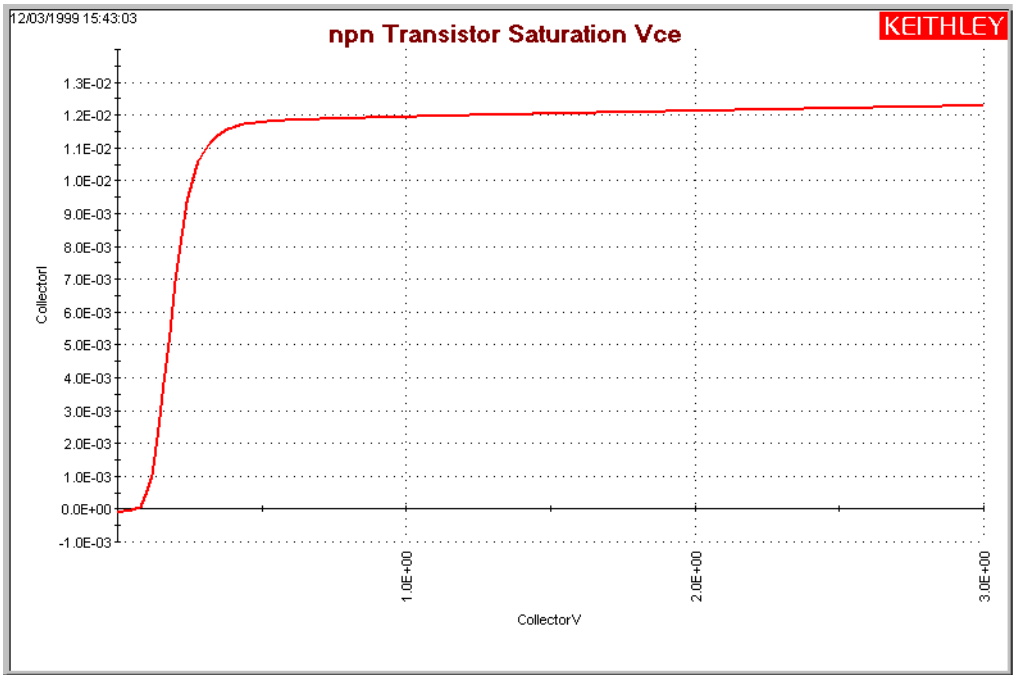
In many cases, you may already know a needed analysis formula, even before running a test. In fact, for an ITM (only) you may create a real-time formula in advance so that you can monitor its output during a test, either in the **Graph** tab or in the **Sheet** tab **Data** worksheet.

However, in other cases, you may decide to perform an analysis after a review of existing test data. The **Formulator** function(s) to use and the data to be included in the **Formulator** calculations must be evaluated to meet your requirements. The next two subsections illustrate one such evaluation.

Determining the type of calculation: an example

For example, suppose, after looking at the BJT saturation voltage plot in Figure 6-361, that you desire to have a better look at the point where the slope of the saturation plateau becomes constant.

Figure 6-361
Example data to be analyzed



You might decide to apply the function `REGFIT` to the **CollectorV** values (and corresponding **CollectorI** values) between 1V and 3V. The line generated by `REGFIT`, when co-plotted with the existing curve, should depart from the plateau at the point of curvature.

The remaining subsections under Configuring Formulator calculations, apply the `REGFIT` function to the data of Figure 6-361 to illustrate use of the **Formulator.**

Determining range data, if required, for a calculation: an example

Many **Formulator** functions do not require you to specify row numbers (indices) as arguments. However, some **Formulator** functions, such as `REGFIT`, require you to specify the range of data to be included in the calculation (typically as the row numbers (indices) for the first and last values to be included (refer to the **Formulator** function reference information, starting under Understanding the Formulator functions later in this section)). This requirement allows you to apply a calculation to only a specific part of the data.

To find the corresponding row numbers (indices) for that specific part of the data, check the **Data** worksheet. For example, referring to Figure 6-362, you might decide to apply `REGFIT` only to the **CollectorV** values between 1V and 3V. Looking at the **Data** worksheet for this data, you note that the **CollectorV** values between 1V and 3V are located between rows (indices) 52 and 152. This is the range information that you need to create the regression analysis equation using `REGFIT`.

Figure 6-362
Determining the starting and ending row numbers (indices) for the data to be analyzed

	A	B	C
1	CollectorI	CollectorV	BaseV
50	1.19339E-02	9.60000E-01	7.53306E-01
51	1.19386E-02	9.80000E-01	7.53295E-01
52	1.19440E-02	1.00000E+00	7.53286E-01
53	1.19477E-02	1.02000E+00	7.53271E-01
54	1.19523E-02	1.04000E+00	7.53268E-01
55	1.19557E-02	1.06000E+00	7.53248E-01
56	1.19601E-02	1.08000E+00	7.53239E-01
57	1.19643E-02	1.10000E+00	7.53218E-01
58	1.19684E-02	1.12000E+00	7.53212E-01
59	1.19721E-02	1.14000E+00	7.53192E-01
60	1.19753E-02	1.16000E+00	7.53178E-01
61	1.19788E-02	1.18000E+00	7.53170E-01
62	1.19839E-02	1.20000E+00	7.53157E-01
63	1.19884E-02	1.22000E+00	7.53157E-01
64	1.19919E-02	1.24000E+00	7.53121E-01
65	1.19948E-02	1.26000E+00	7.53131E-01
66	1.19983E-02	1.28000E+00	7.53110E-01
67	1.20019E-02	1.30000E+00	7.53086E-01
68	1.20059E-02	1.32000E+00	7.53079E-01
69	1.20090E-02	1.34000E+00	7.53060E-01
70	1.20135E-02	1.36000E+00	7.53058E-01
71	1.20185E-02	1.38000E+00	7.53042E-01
72	1.20231E-02	1.40000E+00	7.53025E-01
73	1.20252E-02	1.42000E+00	7.53021E-01
74	1.20296E-02	1.44000E+00	7.52996E-01

Data Calc Settings

	A	B	C
1	CollectorI	CollectorV	BaseV
128	1.22095E-02	2.52000E+00	7.52108E-01
129	1.22134E-02	2.54000E+00	7.52075E-01
130	1.22172E-02	2.56000E+00	7.52071E-01
131	1.22188E-02	2.58000E+00	7.52054E-01
132	1.22226E-02	2.60000E+00	7.52037E-01
133	1.22251E-02	2.62000E+00	7.52005E-01
134	1.22284E-02	2.64000E+00	7.51997E-01
135	1.22318E-02	2.66000E+00	7.51966E-01
136	1.22349E-02	2.68000E+00	7.51951E-01
137	1.22390E-02	2.70000E+00	7.51936E-01
138	1.22413E-02	2.72000E+00	7.51917E-01
139	1.22447E-02	2.74000E+00	7.51894E-01
140	1.22481E-02	2.76000E+00	7.51875E-01
141	1.22512E-02	2.78000E+00	7.51852E-01
142	1.22551E-02	2.80000E+00	7.51830E-01
143	1.22579E-02	2.82000E+00	7.51819E-01
144	1.22602E-02	2.84000E+00	7.51795E-01
145	1.22636E-02	2.86000E+00	7.51776E-01
146	1.22670E-02	2.88000E+00	7.51760E-01
147	1.22703E-02	2.90000E+00	7.51735E-01
148	1.22729E-02	2.92000E+00	7.51729E-01
149	1.22738E-02	2.94000E+00	7.51693E-01
150	1.22787E-02	2.96000E+00	7.51674E-01
151	1.22816E-02	2.98000E+00	7.51653E-01
152	1.22852E-02	3.00000E+00	7.51637E-01

Data Calc Settings

Creating an analysis formula

After you have identified the needed Formulator function(s) and data, create an analysis formula as follows:

1. Enter the left side of the equation. Either use the **F=** assignment or type in a variable name that contains no spaces and is not the same as a **Functions** name.

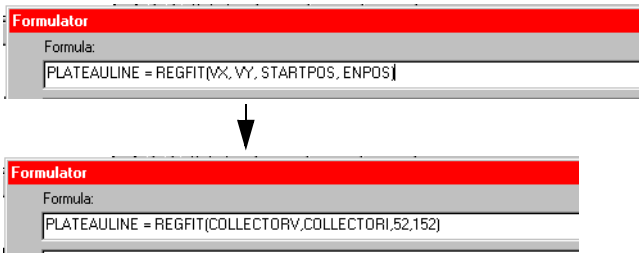
NOTE *Each time that you create an equation with **F=** as the left side, the **formulator** adds a sequential numerical suffix to the **F** when you click the **Add** button. That is, the left side of the first such equation becomes **F1 =**, the left side of the second is **F2 =**, and so on.*

---

2. Enter the right side of the equation at using the function buttons, constant buttons, columns buttons, and keyboard, as appropriate.
   - To insert a function or operator, click a button in the **Functions** area.
   - To replace the format version of an argument in a function (for example, `V1`) with a column (vector) or value from the **Columns** area, select the area in the formula and click the **Columns** item.
   - To insert a constant from the **Constants** area, click the button next to the constant.

For example, to find the regression line for the plateau in Figure 6-361, enter the equation in Figure 6-363.

Figure 6-363
Creating the regression formula for the data



Adding an analysis formula to the ITM or UTM

To move from the upper **Formula** box and enter it in the collection of formulas in the lower **Formula** box, click the **Add** button. See example results in Figure 6-364. If you have previously run the ITM or UTM, this action also executes the new formula for the existing data.

Figure 6-364
Added and executed regression formula for the plateau

To enter an edited formula in the upper box, also click the **Add** button. You are given the option to replace the same-named formula in the lower box or to rename and add it to the collection of formulas. Refer also to Editing formulas and constants.

Executing an analysis formula

If you specify future project data as arguments of a formula (for example, you create the formula when you configure the associated ITM or UTM, before running it) the following occurs:

- If you compose the formula using exclusively real-time functions, it executes in real time during each run of a test.

NOTE *Real-time calculations do not apply to UTM data. A UTM **Data** worksheet does not update until the test is finished.*

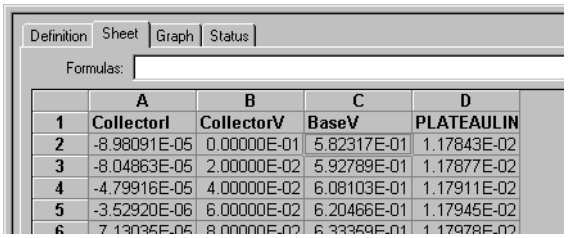
- If you compose a formula containing one or more post test only functions, it executes at the end of each run of the ITM or UTM.

If you specify existing project data as the arguments of a formula, the formula immediately executes and acts on the existing data when you click the **Add** button. Thereafter, it executes as indicated in the bulleted list above.

Viewing analysis results in the Sheet tab Data worksheet

After executing a new formula, a new column of data, containing the results is added to the **Data** worksheet of the **Sheet** tab. See the example in Figure 6-365. If the formula in the upper **Formula** box was edited to replace a prior version, the corresponding **Data** worksheet column updates to reflect the changes.

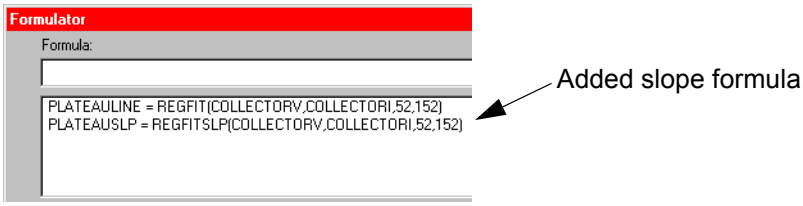
Figure 6-365
Linear regression line for the plateau added to the Data worksheet of the Sheet tab (in column D)



	A	B	C	D
1	CollectorI	CollectorV	BaseV	PLATEAULIN
2	-8.98091E-05	0.00000E-01	5.82317E-01	1.17843E-02
3	-8.04863E-05	2.00000E-02	5.92789E-01	1.17877E-02
4	-4.79916E-05	4.00000E-02	6.08103E-01	1.17911E-02
5	-3.52920E-06	6.00000E-02	6.20466E-01	1.17945E-02
6	7.13035E-06	8.00000E-02	6.33369E-01	1.17978E-02

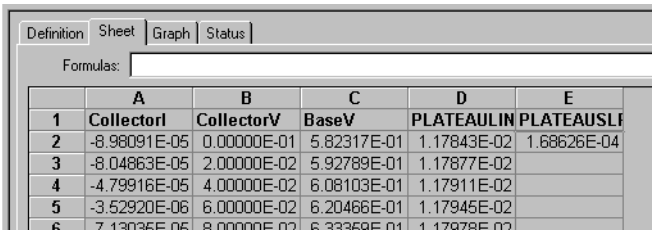
In some cases, a results column contains only a single value. Suppose, for example, that you also added a formula to find the regression slope for the data as illustrated in Figure 6-366.

Figure 6-366
Added linear regression slope formula for the plateau



Then after executing the added slope formula, a fifth column containing only a single number (the slope of the linear regression curve) is added to the **Data** worksheet of the **Sheet** tab. See Figure 6-367.

Figure 6-367
Added linear regression slope result in column E for the plateau



	A	B	C	D	E
1	CollectorI	CollectorV	BaseV	PLATEAULIN	PLATEAUSLI
2	-8.98091E-05	0.00000E-01	5.82317E-01	1.17843E-02	1.68626E-04
3	-8.04863E-05	2.00000E-02	5.92789E-01	1.17877E-02	
4	-4.79916E-05	4.00000E-02	6.08103E-01	1.17911E-02	
5	-3.52920E-06	6.00000E-02	6.20466E-01	1.17945E-02	
6	7.13035E-06	8.00000E-02	6.33369E-01	1.17978E-02	

NOTE *After some **Formulator** calculations, you may see one or more instances of the **#REF** notation in a column, instead of a number. **#REF** in a cell indicates that a valid value could not be calculated. This occurs when a **Formulator** function needs multiple rows as arguments, when a calculated value is out of range, when a divide by zero is attempted, and so on.*

For example, each result of the `DIFF` *function is a difference coefficient that is calculated as the ratio **ΔValues1** /**ΔValues2**, where **ΔValues1** and **ΔValues2** are differences between values in the present row and values in the previous row. Because no previous*

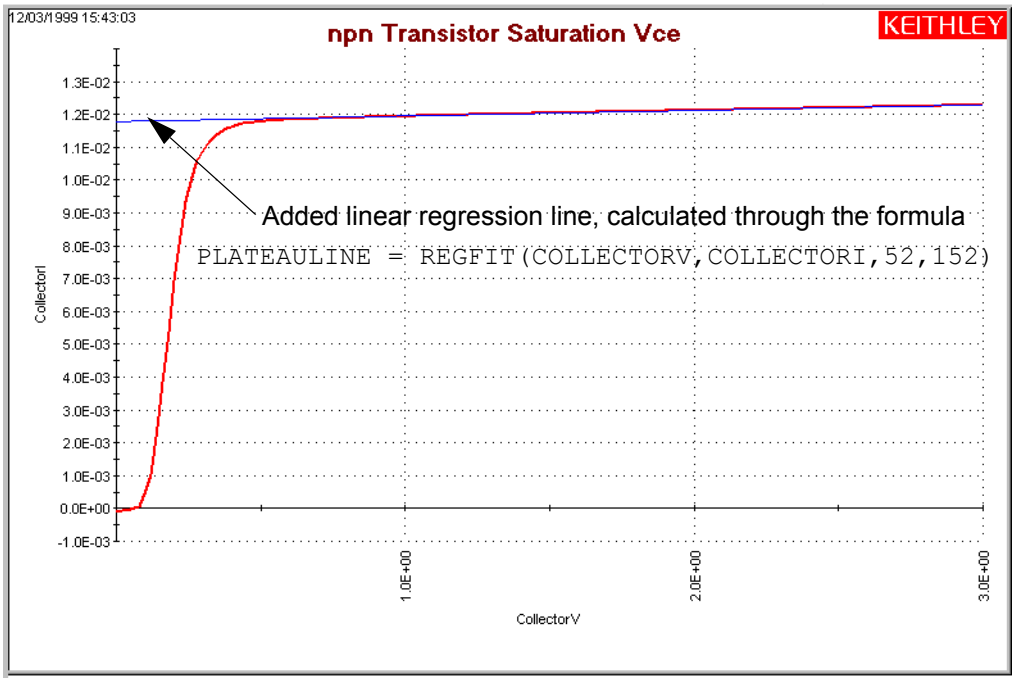
*row exists before the first row, a valid calculation is not possible for the first row. Therefore, the **Formulator** returns the **#REF** notation in the first row.*

*A column will contain multiple instances of **#REF** if the **Formulator** function requires multiple prior cells for the calculation. For example, if the MAVG function is using five data points to calculate a moving average of a column containing five values, the first two and last two cells will contain **#REF**.*

Viewing analysis results in the Graph tab

If a new column (vector) is added to the **Sheet** tab **Data** worksheet after creating/modifying a formula, it can be plotted in the **Graph** tab, just as any other column (vector). See Figure 6-368.

Figure 6-368
Added linear regression line to the graph shown in Figure 6-361.



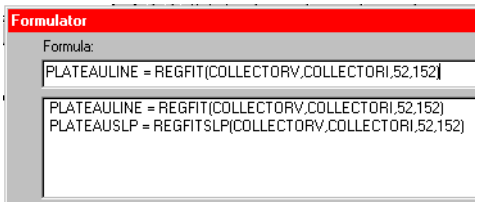
NOTE *For information about using the **Graph** tab, refer to Displaying and analyzing data using the Sheet tab.*

Editing formulas and constants

To Edit a **Formulator** formula, do as follows:

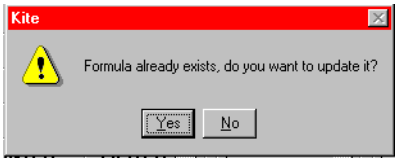
1. In the lower **Formula** box double-click the formula that you wish to edit. A copy of the formula appears in the upper **Formula** box. See the example in Figure 6-369.

Figure 6-369
Editing the linear regression line formula for the plateau



2. In the upper **Formula** box, edit the formula as needed.
3. Click the **Add** button.
   - If you renamed the result variable on the left side of the formula, the **Formulator** adds the edited formula to the lower **Formula** box as a new formula.
   - If you did not rename the variable on the left side of the formula, the **Formulator** assumes that you only wish to update an existing formula. However, it checks to make sure, using the message box of Figure 6-370.

Figure 6-370
Formulator edit message box



4. Respond to the message box per Figure 6-370 according to your needs:
   - If you edited the formula so as to create an additional new formula (but forgot to change the variable name for the result), click **No**. Nothing happens to either of the formula boxes. Edit the name of the result variable, then click **Add** again.
   - If you edited the formula to update it, click **Yes**. The replacement formula appears in the lower **Formula** box. Figure 6-371 shows an (unlikely) modified regression formula that includes all the data in Figure 6-363.

Figure 6-371
Edited-formula illustration

Deleting Formulator formulas and constants

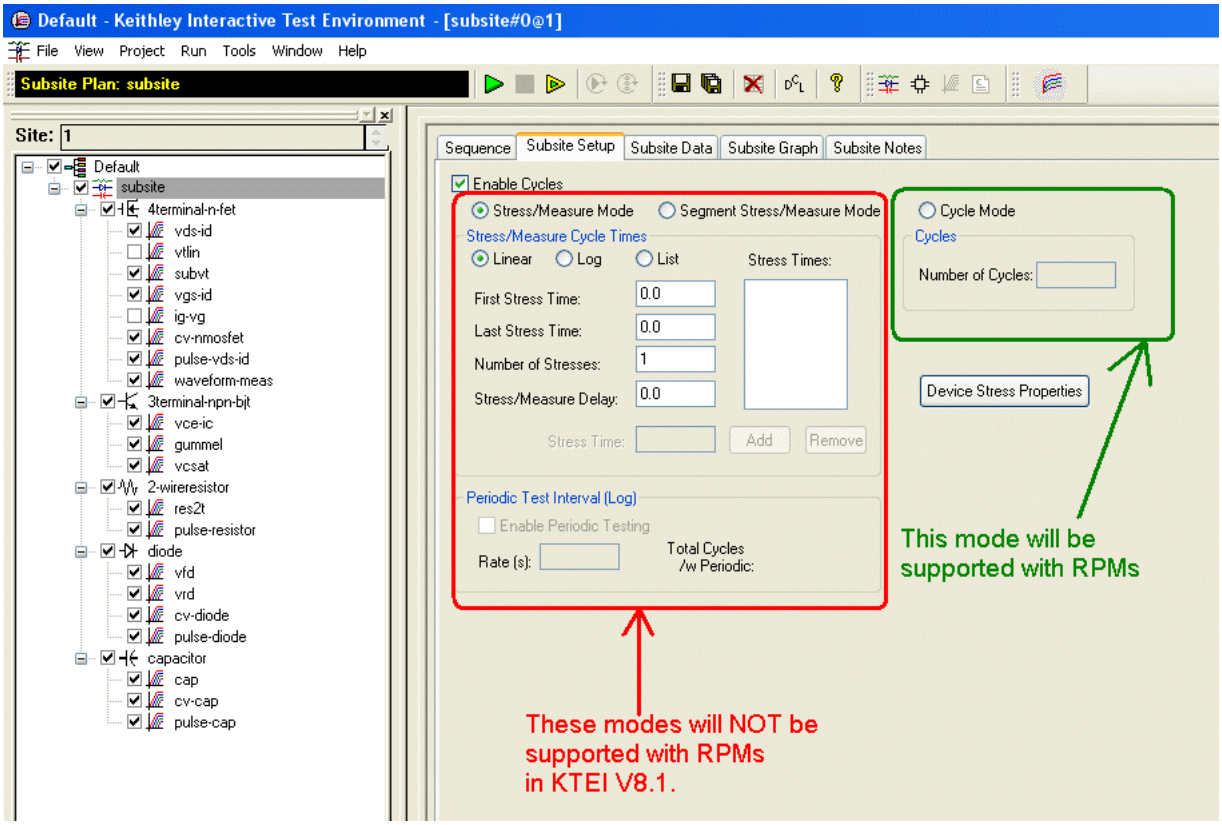
To delete a **Formulator** formula, do as follows:

1. Select the formula with the cursor.
2. Do either of the following:
   - Click the **Delete** button in the Formulator window.
   - Press the **DELETE** key on the keyboard.

Subsite cycling

KITE can perform Hot Carrier Injection (HCI) tests, Negative Bias Temperature Instability (NBTI) tests, and similar Wafer Level Reliability (WLR) tests. The built-in software for stress testing is integrated with subsite cycling. Figure 6-372 shows the KITE tab used to set up subsite cycling.

Figure 6-372
Subsite Setup tab



Overview

Subsite cycling allows you to repeatedly cycle through the subsite tests. The data for every repeated test (ITM or UTM) is acquired and placed in its data Sheet tab. Figure 6-373 shows an example data sheet for a test that was run (cycled) four times. An individual test is opened by double-clicking it in the Project Navigator.

Measured readings can be exported from individual ITMs or UTMs into the Subsite Plan, which has its own data sheet tab and graph tab. For details, see the User's Manual, Subsite cycling data sheets, page 3-87.

There are three subsite cycling modes:

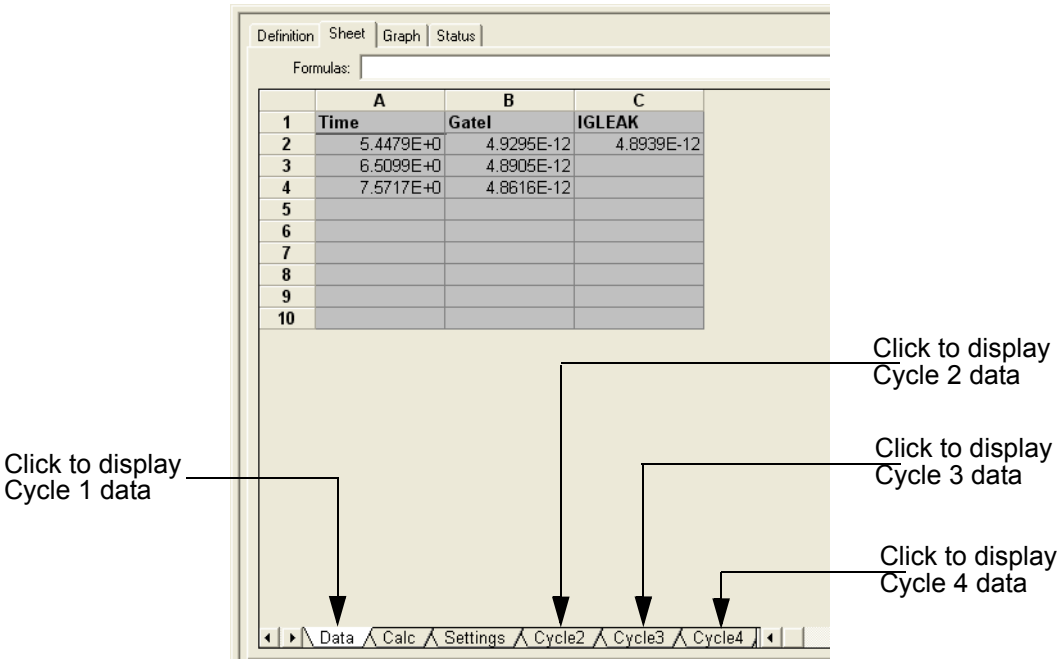
- **Cycle Mode**: This mode performs tests, but does not use device stressing.
- **Stress/Measure Mode**: This mode performs tests and also provides device stressing (DC voltage stress, DC current stress and AC voltage stress).
  - DC stress is applied by one or more SMUs. Devices can be stressed individually, or they can be parallel-connected so that a single SMU can stress multiple devices. The SMUs can also be used to measure the DC stress.

- AC stress is applied by pulse generator cards. Each pulse generator cards has two pulse output channels. Each channel can stress one device terminal.[15]

- **Segment stress/measure mode**: This mode performs tests and also provides device stressing (Segment ARB® waveform stress):
  - Stress is provided by a Segment ARB waveform generated by a pulse generator card. Each channel of the pulse generator card can stress one device terminal.
  - DC bias voltage and current limit for the device can be provided by the SMUs in the system.

NOTE *Only Cycle Mode is supported with Model 4225-RPMs present in the system. The Stress/Measure and Segment Stress/Measure modes are NOT supported if RPMs are present in the system. In order to run Stress/Measure and Segment Stress/Measure modes, all RPMs must be disconnected from the Model 4200-SCS and you must update the RPM configuration in KCON (see Tools > Update DC Preamp and RPM Configuration in Section 7).*

Figure 6-373
Test data example for an individual test: four cycles

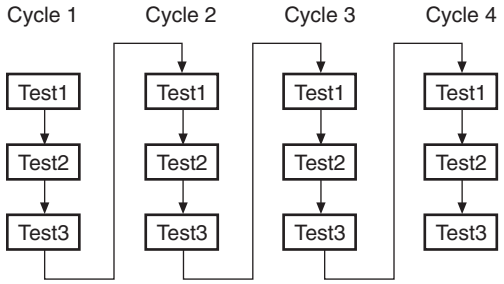


Cycle Mode

In the Cycle Mode, KITE repeatedly cycles through the subsite tests with no stressing. The user specifies the number of cycles to be performed. Assume a test sequence for a subsite plan that is made up of three tests. Figure 6-374 shows a four-cycle sequence for those tests. As shown, each test is run four times.

15. The Models 4205-PG2, 4220-PGU, and 4225-PMU are dual channel pulse cards inside the Model 4200-SCS. To differentiate between the internal pulse cards and other supported external pulse instruments, a Keithley pulse card may also be referred to as a VPU (or Voltage Pulse Unit).

Figure 6-374
Subsite cycling example: four cycles



Perform the following steps to select and set the number of test cycles to perform:

1. In the Project Navigator, double-click the name of the subsite.
2. Click the Subsite Setup tab.
3. Select the **Cycle Mode** and enter the number of cycles to be performed.
4. Click the **Apply** button located at the bottom-right corner of the tab.

Stress/Measure Mode

NOTE *See Stress/Measure Mode for details on using this stress/measure mode.*

The test sequence for the Stress/Measure Mode is similar to the test sequence for the Cycle Mode, but includes components for stressing, % change and target evaluation. Figure 6-375 shows an example of the basic testing sequence. The added components for stressing, % change an target evaluation are shown in blue. Measured readings for the individual test data sheets and Subsite Plan data sheet are acquired in the same manner as for the Cycle Mode (see the User's Manual, Subsite cycling data sheets, page 3-87).

As shown in Figure 6-375, the first cycle runs all tests with no stressing. At the start of the 2nd cycle, all the devices in the Subsite Plan are stressed with voltage or current for a specified period of time. After the stress period expires, stressing is stopped and the three tests are run. Notice that after each test is run, the % or Abs (absolute) Change and Targets are evaluated.

% Change: A post-stress Output Value reading is compared to its pre-stress Output Value reading. The % Change is calculated and listed in the Subsite Plan data sheet. The % Change for post-stressed Output Values are calculated as follows:

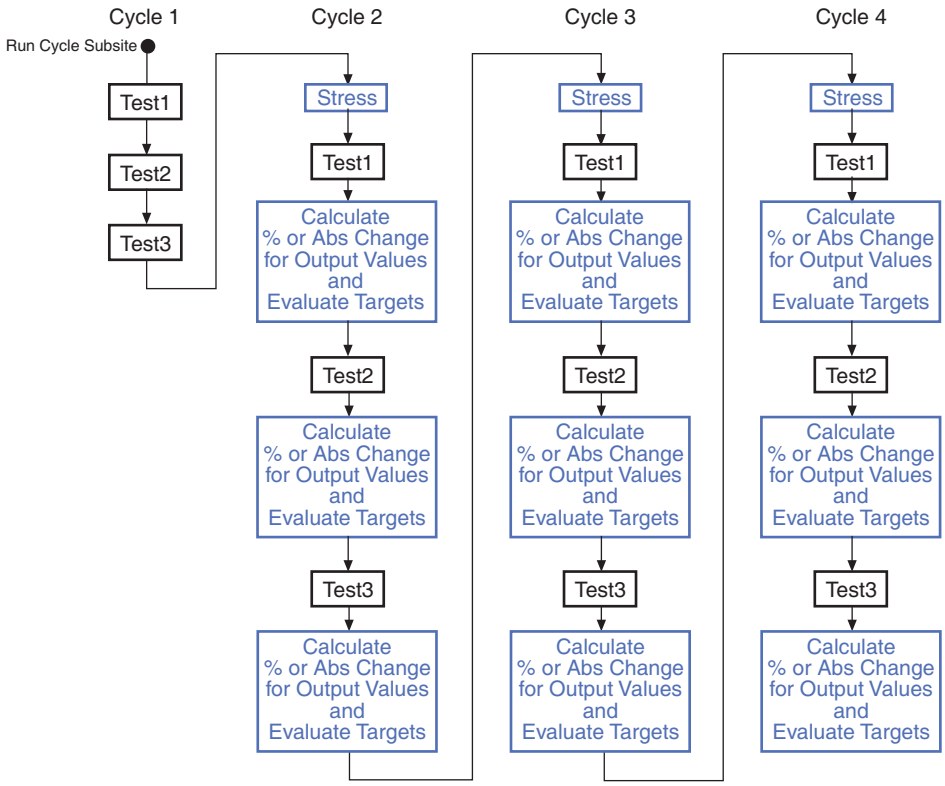
% Change = ABS[(Post-Stress Reading: Pre-Stress Reading) / Pre-Stress Reading x 100]

Abs Change: A post-stress Output Value reading is compared to its pre-stress Output Value reading. The Abs (absolute) Change is calculated and listed in the Subsite Plan data sheet. The Abs Change for post-stressed Output Values are calculated as follows:

Abs Change = ABS[Post-Stress Reading: Pre-Stress Reading]

Segment Stress/Measure Mode

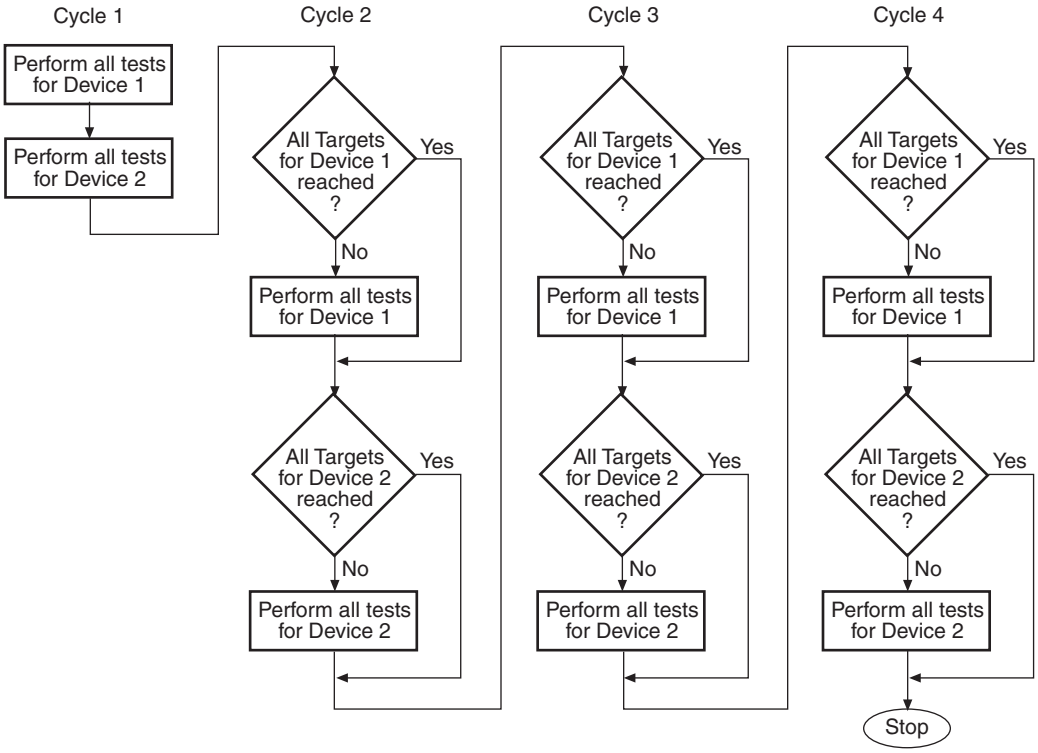
Figure 6-375
Example of the stress testing sequence (four cycles) for a single device



Degradation Targets: Output Values can be enabled as Targets. A Target is specified as a percentage (%) or an absolute value and is evaluated for degradation. When all Targets for a device are reached, then that device will no longer be tested. Subsequent cycles will bypass the device tests that reached all its Targets.

The testing process for Target evaluation is explained by the flowchart shown in Figure 6-376. As a simple example, assume all the Targets for both devices are reached after the first cycle of the Subsite Plan. Following the flowchart (Figure 6-376) will show that the tests for cycles 2, 3 and 4 will not be performed. The Subsite Plan will simply stop.

Figure 6-376
Target evaluation process (example for two devices, four cycles)



Segment stress/measure mode

This stress/measure mode is similar to the Stress/Measure Mode, except device stressing is provided by pulse generator cards using the Segment ARB® pulse mode. SMUs can be used to provide bias voltage and current limit for the devices, but cannot be used to measure stress.

Stress/Measure Mode

NOTE *The following information explains stress testing using the Stress/Measure Mode. Stressing is provided by SMUs or Keithley pulse cards or both (using the standard pulse mode for AC stressing).*

Stressing can also be provided by Keithley pulse cards using the Segment ARB pulse mode. Refer to Segment Stress/Measure Mode for supplemental information on using Segment ARB for stress testing.

For stress testing, a KITE evaluation consists of pre-stress tests at a particular subsite, followed by alternate cycles of stressing and retesting. KITE performs these cycles automatically when you select **Stress/Measure Mode** in the **Subsite Setup** tab. During the evaluation, KITE can display intermediate numerical and graphical results and status information. KITE ends the evaluation when the devices degrade beyond specified exit criteria (target degradation) or when the total stressing time reaches a specified maximum, whichever comes first.

DC Voltage stressing

KITE's built-in stress algorithm uses SMUs to DC voltage stress multiple devices concurrently. The following capabilities apply to device stressing during Hot Carrier Injection studies. Similar capabilities apply to other types of voltage stress-measure studies.

- A unique gate-stress bias voltage (Vg Stress) and a unique drain-stress bias voltage (Vd Stress) can be applied to each evaluated device, within the source limitations of the system. Each unique gate or drain stress bias condition requires a dedicated source. For example, if your Model 4200-SCS system contains four SMUs, you can apply a maximum of four unique stress bias voltages (gate voltages plus drain voltages combined). If your Model 4200-SCS system is fully configured (eight SMUs, four medium power and four high power) you can apply a maximum of eight unique stress bias voltages.
- When some of the devices are connected in parallel, the program can voltage stress up to twenty devices at once, subject to system resource and matrix limitations. Figure 6-377A illustrates a voltage stressing configuration that uses the maximum software and system capabilities.
- If your voltage stress system is using a switch matrix, the Model 4200-SCS will try to maximize the amount of SMU sharing in order to allow parallel testing. It determines what pins can share SMUs in the following fashion. If pins from different devices have the same name (for example, Gate Pin, Drain Pin, etc). and the like named pins are assigned the same voltage stress, then when the stress is applied these pins will all be automatically connected to the same SMU through the switch matrix. That SMU will supply the voltage stress to all the pins simultaneously.
- Because parallel-connected devices share resources, the KITE monitors stressing resources when Device Stress Properties are configured. If the requirement exceeds the resources, KITE reports an error in the **Check Resources** window condition.

AC Voltage stressing

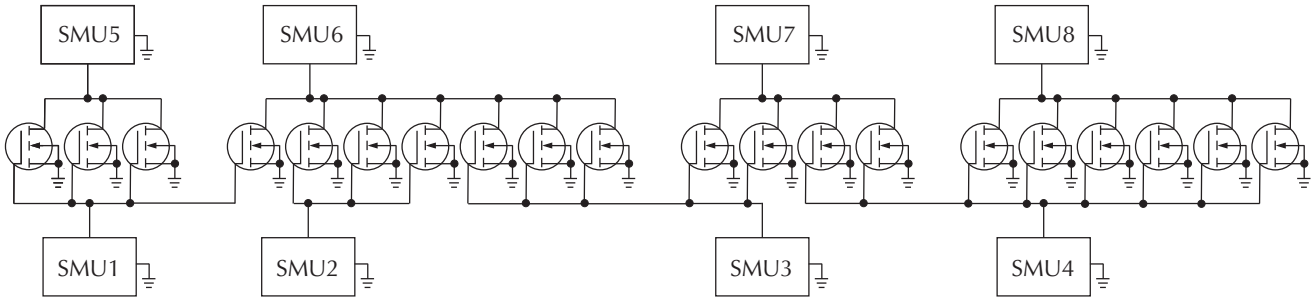
Four Keithley pulse cards can be used to AC voltage stress eight devices (one device pin per pulse channel). Figure 6-377B shows a Keithley pulse card providing AC voltage stress for six devices.

Parallel-connected devices cannot be AC voltage stressed using the pulse card. As shown in Figure 6-377B, each pulse card channel can only stress one pin of one device.

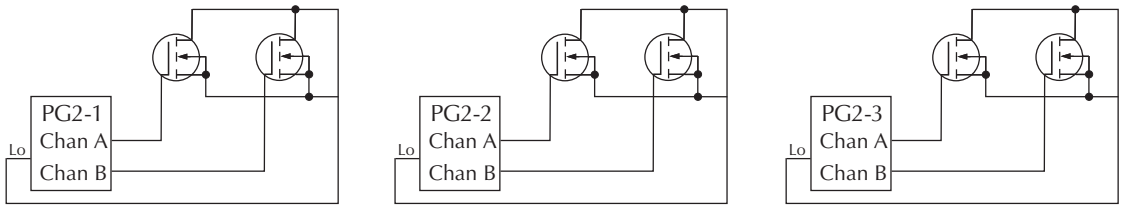
A switch matrix is supported for a pulse card AC voltage stress/measure system. Figure 6-378 shows the use of a switch matrix for an AC/DC voltage stress/measure system. Recommended matrices and cards for this system configuration are the Models 707A/708A and 7174A/7173-50 respectively.

NOTE *To effectively transmit the higher frequency components of the typical pulse (Segment ARB or Standard), a high bandwidth switch matrix should be used (for example, Keithley Instruments Models 7174A or 7173-50).*

Figure 6-377
Voltage stressing (DC and AC)

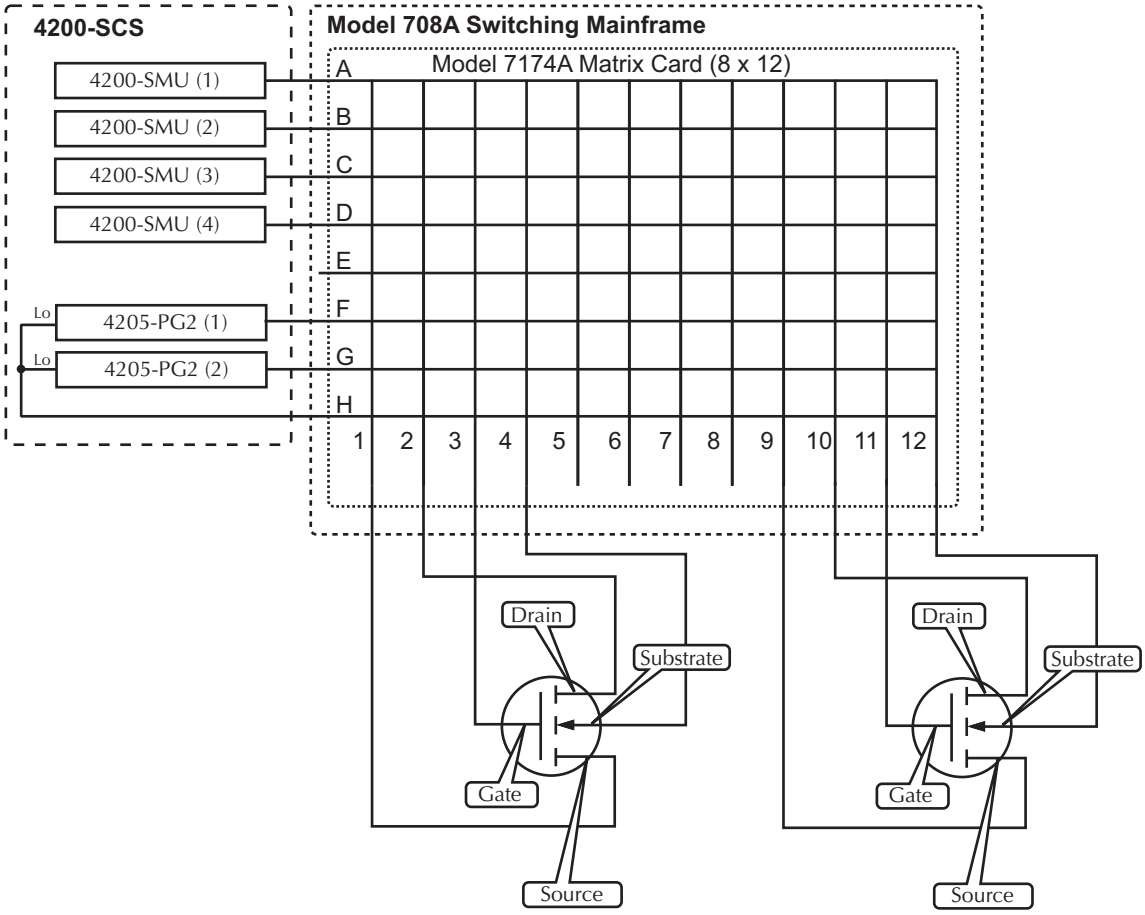


A. DC Voltage Stressing: 20 parallel-connected devices being stressed at eight gate and drain voltages



B. AC Voltage Stressing: Six devices being stressed at the gates using the six pulse outputs of three Model 4205-PG2s

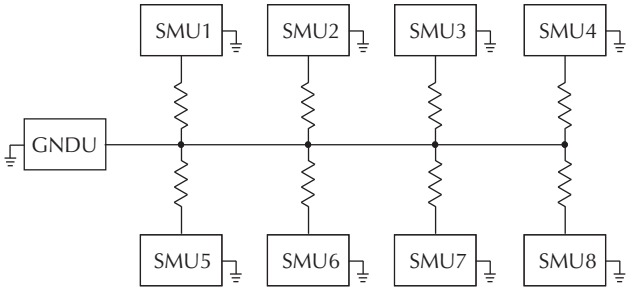
Figure 6-378
Switch matrix for an AC/DC voltage stress/measure system



Current stressing

For current stressing, the maximum number of devices depend on the number of SMUs in the system. Each SMU can current stress one device. For an eight-SMU system, up to eight devices can be current stressed (see Figure 6-391).

Figure 6-379
Eight devices being current stressed by eight SMUs



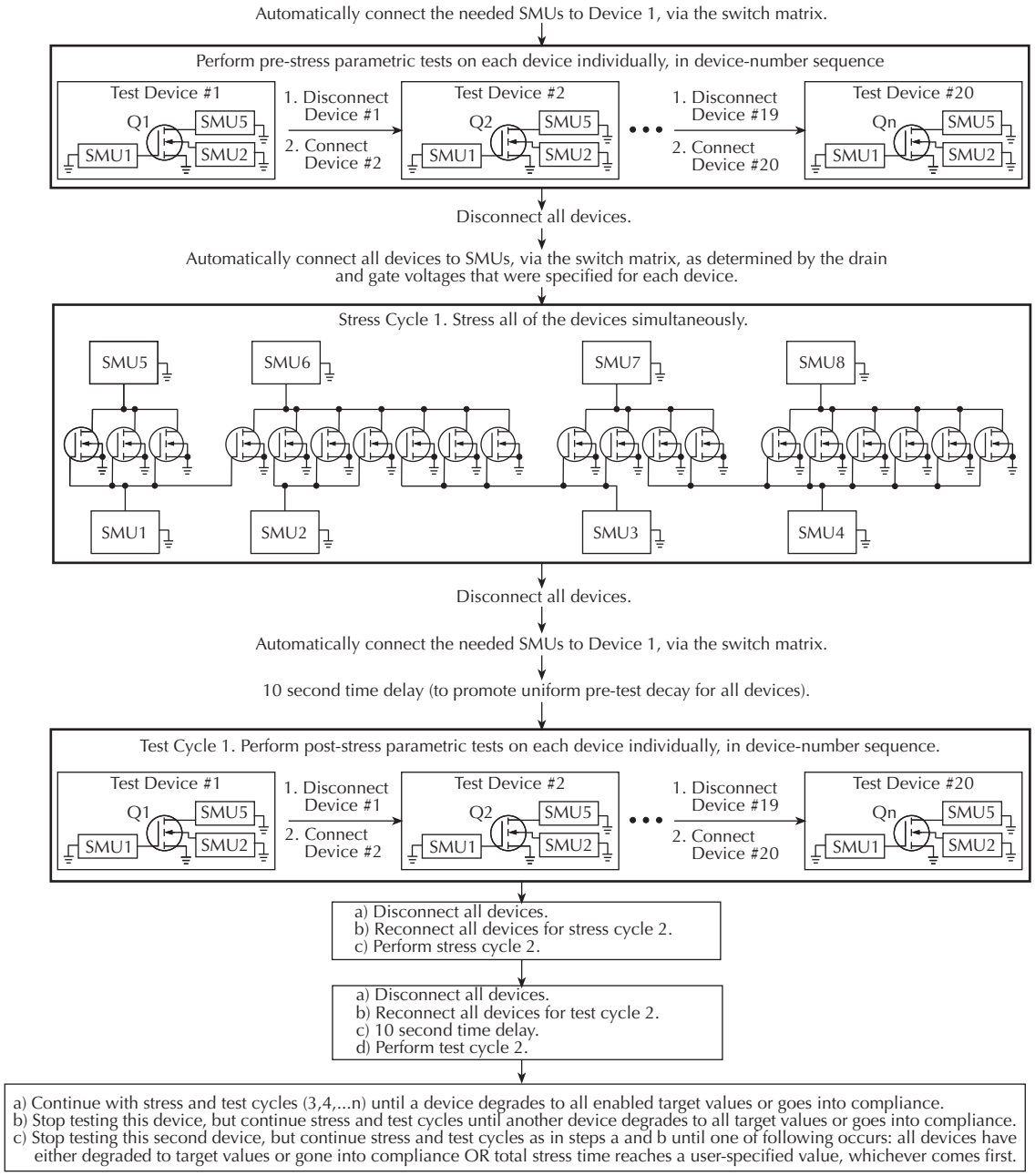
Testing

If you choose **Stress/Measure Mode** during subsite setup, the KITE subsite cycling routine executes tests for the entire subsite in the usual way (as described in previously in this section): initially, before the first stress interval, and then following each stress. If you choose **Cycle Mode** during subsite setup, KITE repeats tests for the entire subsite *without* intermediate stressing, for a user-specified number of iterations.

Combined stressing and testing

Figure 6-380 summarizes an HCI evaluation, for the stressing configuration shown in Figure 6-377A. Similar operations apply to other types of stress-measure studies.

Figure 6-380
Example of combined stressing and testing



NOTE *For information about Reliability Stress Measure testing, refer to Section 3 of the user's manual How to perform reliability (stress-measure) tests on my device.*

NOTE *For information about AC stress for WLR, refer to Section 3 of the user's manual on How to perform AC stress for wafer level reliability (WLR).*

Storing test results in exportable Keithley Data File (KDF) format

You can save all of the data in a KITE project into a single ASCII-formatted file in Keithley Instruments Data File (KDF) format. KDF format is the data format used by KTE in the Keithley Instruments Models S600/630, S400/450, and S900 testers.

Understanding KDF files

Figure 6-381 and Table 6-16 show and describe the possible types and order of KDF entries. Thereafter, Figure 6-382 shows excerpts from a typical Model 4200-SCS KDF. Note that some of the header fields are null or unused, because the Model 4200-SCS is not a multi-lot, multi-wafer system like the Models S600/630/680, S400/450, and S900.

KDF user tag data

The `<TAG>"x,y",value,value` is used to hold the current site's x and y position. The same information is displayed in the Settings Sheet of the ITM or UTM in the Site Coordinates row. This `<TAG>` is inserted before the data for each test. Since each test can have different site coordinates, using `<TAG>` is a way to place the info in the file more than just at the beginning of the site (site_id, row, column). The `<TAG>` is from the Model S600 definition and is used to hold data in the following format:

```
<TAG>"x,y",value,value
```

- `<TAG>` Field identifier.
- `"x,y"` String to identify x and y coordinates.
- `value, value` String that holds the x and y coordinates (for example, 100, 200).

Figure 6-381
Possible types of entries in a Keithley Data File (KDF)

TYP, *file_typ*
LOT, *lot_name*
PRC, *process_name*
DEV, *device_name*
TST, *test_name*
SYS, *system_name*
TSN, *test_station_id_string*
OPR, *operator_name_string*
STT, *dd,mmm,yyyy, tt:tt*
SK1, *usr_data_1*
SK2, *usr_data_2*
SK3, *usr_data_3*
LMT, *limit_file_name*
WDF, *wafer_description_file_name*
COM, *comment_string*
<EOH>
wafer_id,wafer_split,wafer_boat,wafer_slot
site_id,row,column
<TAG>"x,y",value,value
param_id,value
<EOS>
site_id,row,column
<TAG>"x,y",value,value
param_id,value
<EOS>
<EOW>
wafer_id,wafer_split,wafer_boat,wafer_slot
site id,row,column
<TAG>"x,y",value,value
param_id,value
<EOS>
site_id,row,column
<TAG>"x,y",value,value
param_id,value
<EOS>
<EOW>

Color Coding
Red: Header entries
Blue: Wafer-level entries
Olive: Site-level entries

Table 6-16
Descriptions of possible entries in a Keithley Data File (KDF)

Type of entry	Entry	Description		Present in a 4200 KDF?
Header	TYP, file_typ	The type of data file. Typically `TYP, KDF Vn.n` where `n.n` is the version number.		●
	LOT, lot_name	The name of the lot from which the test wafer was selected. String may contain up to 50 characters.		●
	PRC, process_name	The name of the process that produced the lot. String may contain up to 50 characters.		
	DEV, device_name	Device name. String may contain up to 50 characters.		
	TST, test_name	The test name. String may contain up to 255 characters.		●
	SYS, system_name	The system name. String may contain up to 20 characters.		●
	TSN, test_station_id_string	The test station ID integer (1-4).		
	OPR, operator_name_string	The operator name. String may contain up to 30 characters.		●
	STT, dd,mmm,yyyy, tt:tt	Date and time the file was created in Y2K-compliant form.		●
	SK1, usr_data_1	User search key. String may contain up to 30 characters.		
	SK2, usr_data_2	User search key. String may contain up to 20 characters.		
	SK3, usr_data_3	User search key. String may contain up to 10 characters.		
	LMT, limit_file_name	Parameter limits file (`.klf`) name. String may contain up to 80 characters.		
	WDF, wafer_description_file_name	Wafer Description File name. String may contain up to 80 characters.		
	COM, comment_string	Any relevant text, up to 256 characters.		●
	<EOH>	End-of-header tag. Terminates the header.		●
Wafer	wafer_id,wafer_split,wafer_boat,wafer_slot	The wafer ID and (optionally) the wafer split, carrier (boat), and carrier slot during production.	NOTE: Each comma-separated entry is a one-word string that must be a valid C-style identifier: it must start with a nonnumeric character and may contain only letters (a-z, A-Z), digits (0-9), and the underscore character (_).	●
Site	site_id,row,column	The ID and location of the tested site (row and column).	NOTE: Site ID must be a one-word string and a valid C-style identifier (see above).	●
	<TAG>"x,y",value,value param_id,value ... param_id,value	Parameter ID and measured value, one pair per line, for each measurement that was performed at the specified site.	NOTE: In a S400/600/900 KDF, each parameter ID must be a one-word string and a valid C-style identifier (see above). The identifier matches a parameter ID in the parameter limits file (`.klf`).	● Different format. See Figure 6-382.
	<EOS>	End-of-site tag. An <EOS> terminates the data entries for a site section (a KDF section starting with site_id,row,column).		●
Wafer	<EOW>	End-of-wafer tag. An <EOW> terminates the entries for a wafer section (a KDF section starting with wafer_id,wafer_split,wafer_boat,wafer_slot).		●

In Figure 6-382 below, an abbreviated Model 4200-SCS KDF, the entries are color coded essentially as in Figure 6-381 and Table 6-16 above. However, alternate shades of green (olive and lighter green) are used for the site-level entries to separate the data for one parameter from the data for another.

Figure 6-382
Example of Model 4200-SCS KDF

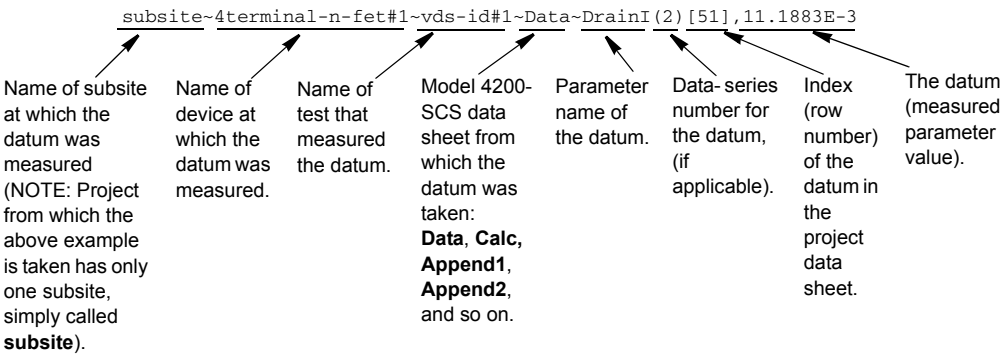
```
TYP,KDF V5.0
LOT,ivswitch
TST,ivswitch
SYS,0123456789
OPR,kiuser
STT,02-Oct-2003 12:01
COM,The format of each data line is as follows: sub-
Site~device~test~sheet_name~column_name[arraySubscript],result
COM,For subsite-level data, ~device~test~ will be null (ie: ~~~)
<EOH>
Wafer_O356,,1,1
0,0,0
<EOS>
1,0,0
<TAG>"x,y",0,0
subsite#0~4terminal-n-fet#1~connect#1~Data~ConnectPins[1],0
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(1)[1],6.4304E-9
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(1)[2],542.2418E-6
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(1)[51],2.2557E-3
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(1)[1],000.0000E-3
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(1)[2],100.0000E-3
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(1)[51],5.0000E+0
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(1)[1],2.0000E+0
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(1)[2],2.0000E+0
     •
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(1)[51],2.0000E+0
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(2)[1],50.6456E-9
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(2)[2],1.0066E-3
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainI(2)[51],11.1883E-3
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(2)[1],000.0000E-3
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(2)[2],100.0000E-3
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~DrainV(2)[51],5.0000E+0
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(2)[1],3.0000E+0
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(2)[2],3.0000E+0
     •
     •
subsite~4terminal-n-fet#1~vds-id#1~Data~GateV(2)[51],3.0000E+0
     •
     •
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[1],2.0000E+0
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[2],1.9500E+0
     •
     •
     •
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[81],-2.0000E+0
subsite~4terminal-n-fet#1~subvt#1~Data~STARTI[1],100.0000E-12
subsite~4terminal-n-fet#1~subvt#1~Data~STOPI[1],100.0000E-6
subsite~4terminal-n-fet#1~subvt#1~Data~IDFIT[1],458.4261E-3
subsite~4terminal-n-fet#1~subvt#1~Data~IDFIT[2],213.7104E-3
     •
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[1],2.0000E+0
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[2],1.9500E+0
     •
```

NOTE: *The data-line format is explained in detail at the bottom of the next page of this example.*

Figure 6-382 (continued)

```
      •
      •
subsite~4terminal-n-fet#1~subvt#1~Data~GateV[81],-2.0000E+0
subsite~4terminal-n-fet#1~subvt#1~Data~STARTI[1],100.0000E-12
subsite~4terminal-n-fet#1~subvt#1~Data~STOPI[1],100.0000E-6
subsite~4terminal-n-fet#1~subvt#1~Data~IDFIT[1],458.4261E-3
subsite~4terminal-n-fet#1~subvt#1~Data~IDFIT[2],213.7104E-3
      •
      •
      •
      •
      •
subsite~4terminal-n-fet#1~subvt#1~Data~IDFIT[81],1.3987E-27
subsite~4terminal-n-fet#1~subvt#1~Data~SUBVTSLP[1],15.2635E+0
subsite~4terminal-n-fet#1~vgs-id#1~Data~DrainI[1],172.1961E-15
subsite~4terminal-n-fet#1~vgs-id#1~Data~DrainI[2],212.1117E-15
      •
      •
      •
subsite~4terminal-n-fet#1~vgs-id#1~Data~DrainI[101],7.7305E-3
subsite~4terminal-n-fet#1~vgs-id#1~Data~GateV[1],000.0000E-3
subsite~4terminal-n-fet#1~vgs-id#1~Data~GateV[2],50.0000E-3
      •
      •
      •
subsite~4terminal-n-fet#1~vgs-id#1~Data~GateV[101],5.0000E+0
subsite~4terminal-n-fet#1~vgs-id#1~Data~GM[1],#REF
subsite~4terminal-n-fet#1~vgs-id#1~Data~GM[2],798.3119E-15
      •
      •
      •
subsite~4terminal-n-fet#1~vgs-id#1~Data~GM[101],1.3701E-3
      •
      •
      •
      •
      •
subsite~3terminal-npn-bjt#1~vce-ic#1~Data~CollectorI(1)[1],-991.2593E-9
subsite~3terminal-npn-bjt#1~vce-ic#1~Data~CollectorI(1)[2],924.5468E-9
      •
      •
      •
subsite~3terminal-npn-bjt#1~vce-ic#1~Data~CollectorI(1)[41],226.4691E-6
      •
      •
      •
      •
      •
subsite~capacitor#1~cap#1~Data~Time[1],000.0000E-3
subsite~capacitor#1~cap#1~Data~Time[2],739.8666E-3
      •
      •
      •
subsite~capacitor#1~cap#1~Data~Time[40],28.8569E+0
<EOS>
<EOW>
```

Data-line format for a Model 4200-SCS KDF, as illustrated above



Note the following general information about KDFs:

- When generating a KDF, KITE ignores data file entries that begin with a # symbol.
- A Model 4200-SCS KDF is compatible with the Model S600/S400-based Keithley Summary Utility (KSU). However, before running a Model 4200-SCS KDF through the KSU, you must convert it from a PC-based ASCII file to a UNIX-based ASCII file, using a utility such as DOS2UNIX. For further information on the KSU, please refer to a Model S600/S400 KTE manual.
- When S400/600/900 series testers generate KDFs automatically during test execution, they store these files in the `$KI_KTXE_KDF` directory.

Generating a KDF

To generate a KDF from new or saved KITE data, do the following:

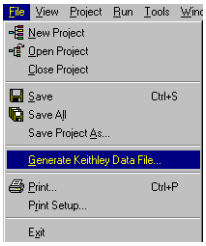
1. If the project that created the data to be KDF-converted is not open, open it now.

NOTE *Illustrations that follow in this procedure typically show KDF generation from data created by the Keithley Instruments-supplied* ***ivswitch*** *project.*

Figure 6-383
ivswitch project

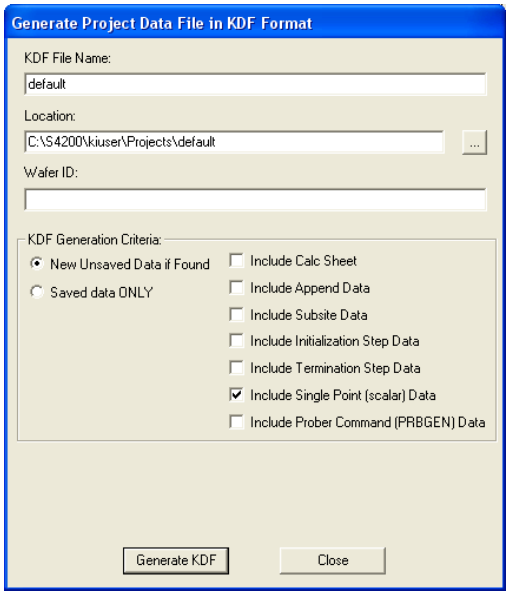
2. In the **File** menu, select **Generate Keithley Data File...**

Figure 6-384
Generating a KDF



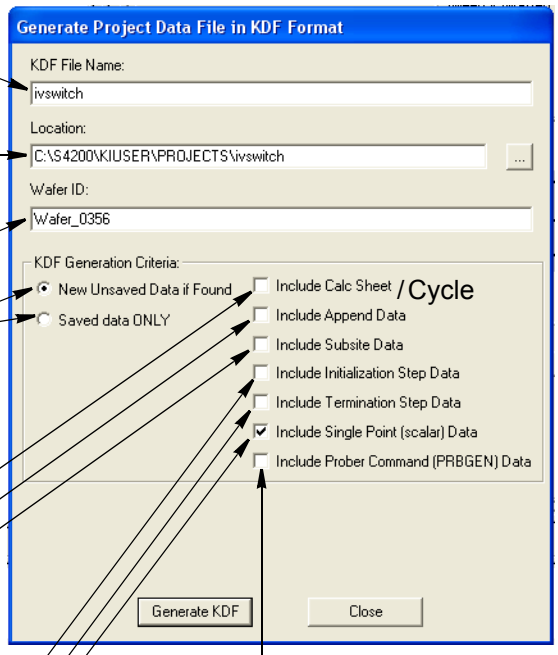
The KDF setup window appears (named Generate Project Data File in KDF Format). See Figure 6-385.

Figure 6-385
KDF setup window



3. In the KDF setup window, enter the required information as explained in Figure 6-386 and the NOTE that follows.

Figure 6-386
Using the KDF setup window



Name for the KDF. The default filename is the same as the name of the presently open project (See above). You can change the name if desired.

Path for the KDF. The default path is the same as the path of the presently open project (See above). You can change the path if desired. However, all folders in the path must already exist: KITE does not create new folders.

Wafer ID (required entry). Maximum string size is 128 characters

The data that is to be converted into a KDF. Select New Unsaved Data if Found (the default) to convert data that is newly created by the project and is not yet saved. Select Saved Data Only to convert data that was saved the last time data was created by the project. See more information in the NOTE that follows this figure.

When the "Include Calc Sheet" checkbox is checked, this instructs the KDF generator to include Calc Sheet data from:
a) All ITMs and all UTMs under devices.
b) All initialization Step UTMs if the "Include Initialization Step Data" checkbox is checked.
c) All Termination Step UTMs if the "Include Termination Step Data" checkbox is checked.
d) All Subsites if the "Include Subsite Data" checkbox is checked.
This checkbox is unchecked by default.

When the "Include Append Data / Cycle Data" checkbox is checked, this instructs the KDF generator to include Append Sheet data from:
a) All ITMs and all UTMs under devices.
b) All initialization Step UTMs if the "Include Initialization Step Data" checkbox is checked.
c) All Termination Step UTMs if the "Include Termination Step Data" checkbox is checked.
AND
Cycle Data from:
d) All ITMs and all UTMs under devices.
This checkbox is unchecked by default.

When the "Include Subsite Data" checkbox is checked, this instructs the KDF generator to include all Subsite Cycle data from all subsites.
This checkbox is unchecked by default.

When the "Include Initialization Step Data" checkbox is checked, this instructs the KDF generator to include all Data Sheet data from all UTMs under the Initialization Step project tree node.

When the "Prober Command (PRBGEN) Data" checkbox is checked, this instructs the KDF generator to include all data returned from the following UTM/User Module calls:
a) PrChuck
b) Prinit
c) PrMovNxt
d) PrSSMovNxt

When the "Include Single Point (scalar) Data" checkbox is checked, this instructs the KDF generator to include all single point data from all checked sources. If this checkbox is unchecked, and data column that has only one data point, will not be included in the KDF file.
This checkbox is checked by default.

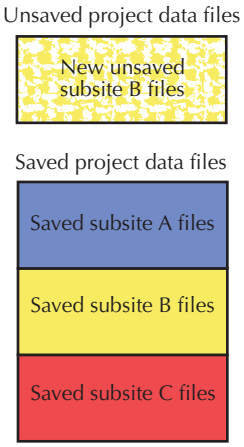
When the "Include Termination Step Data" checkbox is checked, this instructs the KDF generator to include all Data Sheet data from all UTMs under the Termination Step project tree node.

NOTE *If you select **New Unsaved Data if Found**, KITE generates the KDF using the most recent data that has been acquired, even if this data has not yet been saved. If you select "**Saved data ONLY**," KITE generates the KDF using only data that has already been saved.*

For example, suppose a project contains three subsites: A, B, and C. You tested all three subsites in the past and saved the data. Just now, you tested subsite B again (by itself: but have not yet saved the new data. As a result, you presently have two

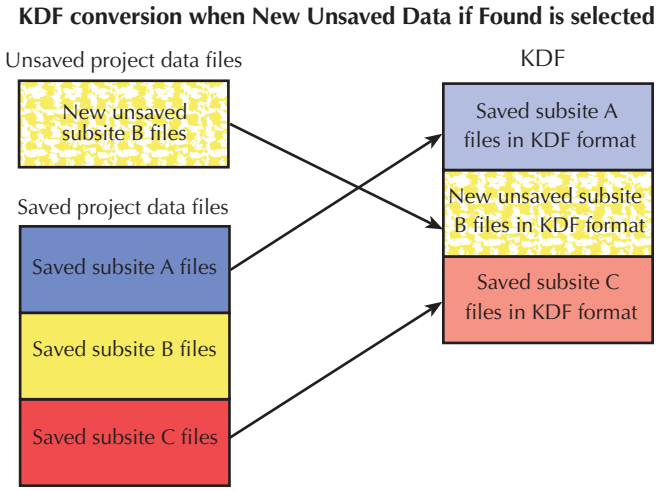
sets of data files) a set of saved files containing data for all three subsites and a set of temporary files containing only unsaved data for subsite B. See the illustration below

Figure 6-387
Unsaved and Saved project data files



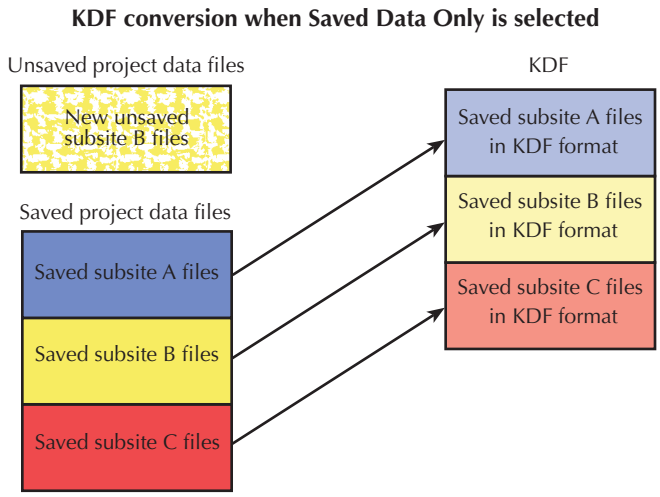
*If you select **New Unsaved Data if Found,** KITE generates the KDF from the unsaved subsite B files and the saved subsite A and subsite C files. See the illustration below.*

Figure 6-388
KDF conversion: New Unsaved Data if Found



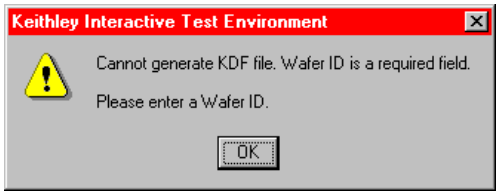
*However, if you select **Saved Data Only,** KITE generates the KDF from the saved subsite A, subsite B, and subsite C files. See the figure below.*

Figure 6-389
KDF conversion: Saved data ONLY



4. Click on the Generate KDF button. If you forget to enter a wafer ID in the Generate Project Data File window, KITE reminds you and inhibits KDF generation. See Figure 6-390.

Figure 6-390
Wafer ID reminder



5. If the message of Figure 6-390 appears, enter the wafer ID and click OK. One of the following occurs:
   - KDF generation begins. Skip to step 7.
   - KDF generation does not begin and KITE cautions you about overwriting KDF data. This occurs if you chose a KDF File Name and Location for which saved data already exists. See Figure 6-391.

Figure 6-391
Overwrite caution

6. If you receive a message similar to Figure 6-391 AND accept overwriting of the existing data, click on Yes (Clicking on No aborts the KDF generation procedure).
7. KDF generation proceeds and KITE displays the file-conversion status, near the bottom of the KDF setup window. See Figure 6-392.

Figure 6-392
Typical file-conversion status display

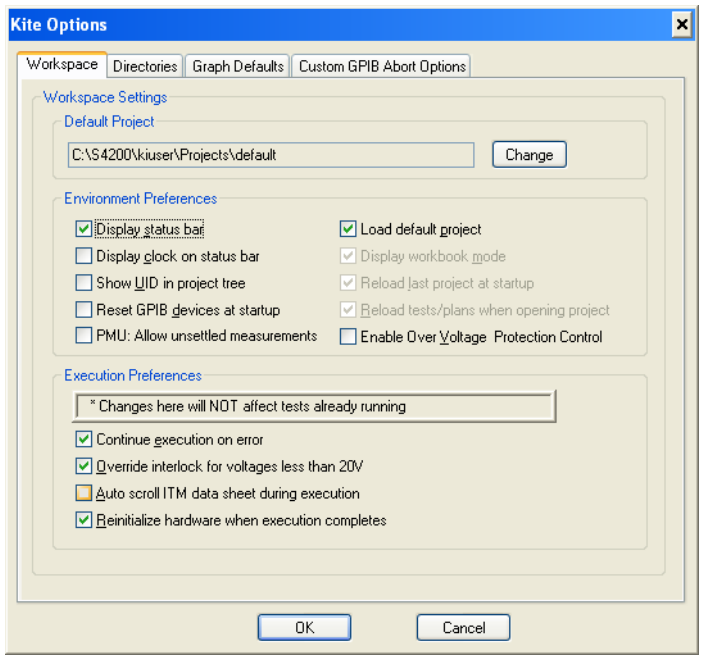
When KDF generation finishes, KITE beeps and displays the following status message near the bottom of the KDF setup window: KDF File Successfully Created.

Customizing KITE

Customizing workspace options

KITE provides several workspace options. These options are accessed through the **Workspace** tab of the KITE Options window. To open the **Workspace** tab, in the KITE **Tools** menu, select **Options**. The KITE Options window opens, displaying the **Workspace** tab by default. See Figure 6-393.

Figure 6-393
Workspace tab of the KITE Options window



The three areas of the Workspace tab are discussed in the next three subsections:

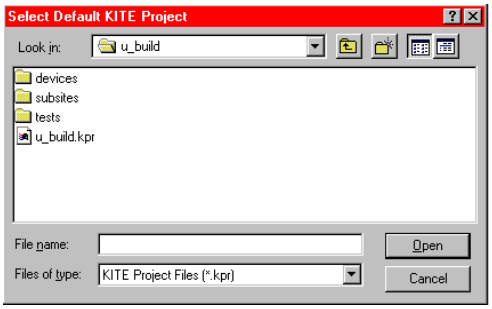
- The **Default Project** area in Specifying a default project
- The **Environment Preferences** area in Specifying environment preferences
- The **Execution Preferences** area in Specifying execution preferences

Specifying a default project

Some situations require that the same project is loaded by default every time a user starts KITE. To specify loading of a specific project by default, do the following:

1. In the **Default Project** area of the **Workspace** tab, click on the **Change** button. A Select Default KITE Project window opens, displaying the directory of the presently open project. See Figure 6-394.

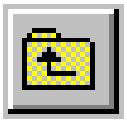
Figure 6-394
Select Default KITE Project window



2. In the **File Name** edit box of the Select Default KITE Project window, enter the **<ProjectName>.kpr** project name using one of the following methods:
   - **Method I**: Type **<ProjectDirectoryPath><ProjectName>.kpr** directly in the **File Name** edit box.
   - **Method II**: Browse for the file name as follows:

   a. Do one of the following:
      - If the project is in the default user directory,[16] then click the next-file-level-up button, illustrated below.

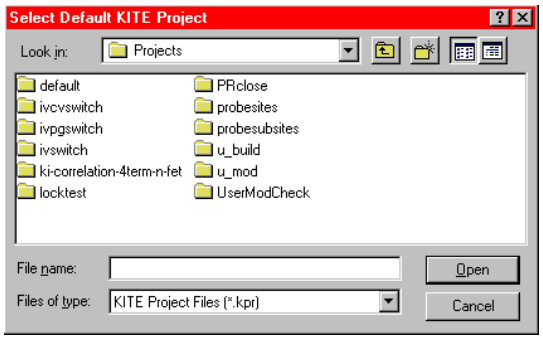
Figure 6-395
Next-file-level-up button



      - If the project is *not* in the default user directory, in the **Look In** combo box, browse for and insert the correct project directory (typically **<Path>\Projects**).

      The Select Default KITE Project window should now display the folders for all project files in the default or otherwise specified project directory. See Figure 6-396.

Figure 6-396
Select Default KITE Project window example showing all projects in directory

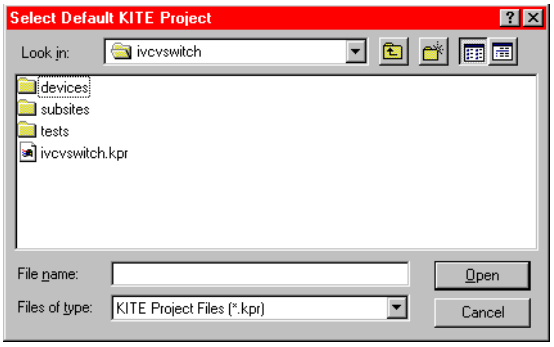


16. For example, the C:\S4200\kiuser\Projects factory default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Projects.

b. Double-click on the **<ProjectName>** folder (the folder that contains the project to be opened). The Select Default KITE Project window displays the file tree for the project to be opened.

For illustration, **ivcvswitch** was selected as the default project, resulting in the file tree shown in Figure 6-397.

Figure 6-397
Select Default KITE Project window: Desired project file tree

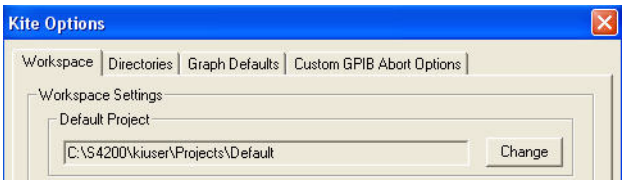


c. Click on the **<ProjectName>.kpr** file name (in our example, **ivcvswitch.kpr).** The file name is entered in the **File name** edit box.

3. In the Select Default KITE Project, click **Open**. The **Default Project** area of the KITE Options window now displays the new default project.

   Figure 6-398 shows the **ivcvswitch** name and file path displayed in the **Default Project** area.

Figure 6-398
Illustration of new default directory



4. In the **Environment Preferences** area of the **Workspace** tab, check the **Load default project** checkbox to instruct KITE to load the specified default project every time a user starts KITE. This action simultaneously checks and denies access to the three **Environmental Preferences** checkboxes that are located below the **Load default project** checkbox.
5. At the bottom of the KITE Options window, click **OK**. The default project will now open every time that you start KITE.

Specifying environment preferences

Each **Environment Preferences** checkbox in the **Workspace** tab is summarized below:

- **Gradient caption**: When **Gradient caption** is checked, the gradient caption bar is displayed above the KITE window as follows:

Figure 6-399
Gradient caption bar

When **Gradient caption** is *not* checked, a standard Windows caption bar is displayed above the KITE window as follows:

Figure 6-400
Windows caption bar

- **Display status bar**: When **Display status bar** is checked, the status bar (see below) is displayed at the bottom of the KITE window.

Figure 6-401
Display status bar

Otherwise, the status bar is not displayed.

- **Display clock on status bar**: When **Display clock on status bar** is checked, the clock on the status bar (see pervious item) is displayed at the bottom of the KITE window. Otherwise, the clock is not displayed.

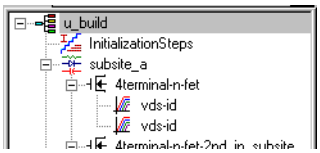
NOTE *On older 233MHz Model 4200-SCS systems, when **Display clock on status bar** is checked, the screen saver for the flat panel display will not activate. Refer to System Administration in Section 10 for more information.*

- **Show UID in project tree**: When **Show UID in project tree** is checked, the Unique IDentifier (UID) numbers are displayed in the Project Navigator (see below). Otherwise, the UID numbers are not displayed.

Figure 6-402
UID numbers in Project Navigator

Show UID in project tree checked

Show UID in project tree unchecked



- **Reset GPIB devices at startup**: If checked, the GPIB devices in the system will reset to their default settings at startup.

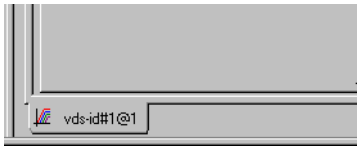
  PMU: Allow unsettled measurements:

  If checked, permits all 4225-PMU instrument cards to ignore the minimum timing versus measure range relationship shown in the chart (Figure 16-45). This is only recommended for advanced users, as spot mean results will not be settled, which may cause a variety of operational issues: inconsistent current measure range-changing on the 4225-PMU or 4225-RPM; lack of proper Load line effect compensation (LLEC) for the PMU and lack of correlation between PMU or PMU+RPM results and SMU results. See PMU minimum settling times versus current measure range for additional information.

- **Load default project**: When **Load default project** is checked, the project that is specified under **Default Project** is loaded every time KITE is started (refer to Specifying a default project).
- **Display workbook mode**: When **Display workbook mode** is checked, window tabs are displayed at the bottoms of project windows, initialization and termination plan windows, subsite-plan windows, device-plan windows, and ITM and UTM windows. Otherwise, the tabs are not displayed. See Figure 6-403.

Figure 6-403
Display workbook mode

ITM window when the Display Workbook Mode box is checked



Same ITM window when the Display Workbook Mode box is unchecked

- **Reload last project at startup**: When **Reload last project at startup** is checked, the project that was open when you exited KITE reopens automatically when you restart KITE.
- **Reload tests/plans when opening project**: When **Reload tests/plans when opening project** is checked, the KITE Workspace interfaces (windows, etc). that were open when you exited KITE reopen automatically when you restart KITE.

Specifying execution preferences

Each **Execution Preferences** checkbox in the **Workspace** tab is summarized below:

- **Continue execution on error**: If **Continue execution on error** is checked, KITE continues executing a project plan sequence when it encounters errors in one or more tests. Examples of erroneous tests include ITMs for which no SMUs have been specified and UTMs that are unconfigured or improperly configured. When KITE encounters the error, it displays an error message in the message area at the bottom of the KITE window and continues execution on the next test in the sequence.
- **Override interlock for voltages less than 20 V**: If **Override interlock for voltages less than 20 V** is checked AND the Model 4200-SCS interlock circuit is disconnected or otherwise open, KITE continues to execute tests. However, KITE automatically limits the output voltage to a safe level, even if a test specifies a higher level.

  If **Override interlock for voltages less than 20 V** is *unchecked* AND the Model 4200-SCS interlock circuit is disconnected or otherwise open, KITE displays a warning message and disables the execution of all tests.
- **Autoscroll ITM data sheet during execution:** If **Autoscroll ITM data sheet during execution** is checked AND **Sheet** tab **Data** worksheet is being viewed in real time during test execution, then KITE scrolls the worksheet such that new data is always visible (this feature is unchecked by default).
- **Reinitialize hardware when execution completes**: If checked, all instruments in the system will return to their default settings after the test is completed.

WARNING ***If "Reinitialize hardware when execution completes" is disabled (unchecked), all outputs will remain at their final levels after the test is completed. Hazardous voltages may be present even though a test is not running.***

Customizing directory options

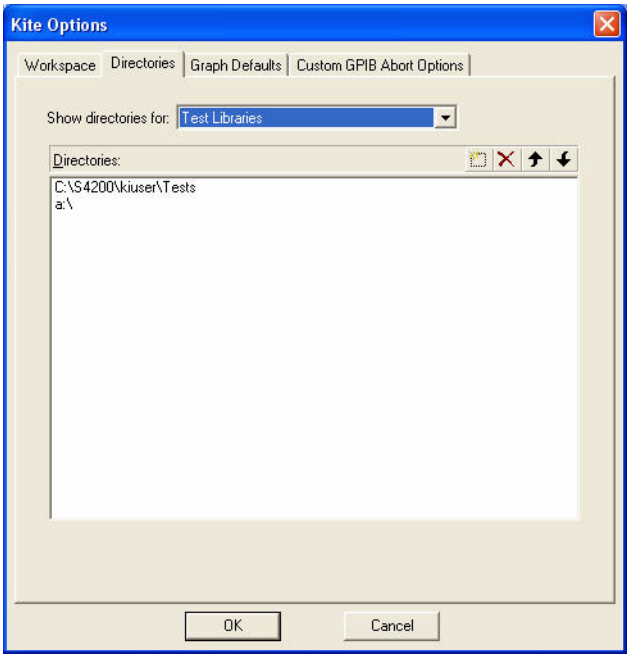
When inserting existing devices, ITMs, and UTMs in a project plan, you can normally choose these only from the default device library and test library directories. The next two subsections explain how to expand your choices, using the **Directories** tab of the KITE Options window.

Open the **Directories** tab as follows:

1. In the KITE **Tools** menu, select **Options**. The KITE Options window opens, displaying the **Workspace** tab by default. See Figure 6-393.

2. In the KITE Options window, click on the **Directories** tab label. See Figure 6-404.

Figure 6-404
Directories tab displaying a default test library



Specifying which test library directories are to be available to projects

When choosing project ITMs and UTMs through a Device Plan window, you can normally choose these only from the test library that resides in the default user directory.[17] However, if you previously submitted tests to a library other than the default test library (as directed in Submitting tests to a library), then you may desire to choose tests from multiple libraries. Therefore, KITE allows you to add (and delete) other test library selections to all Device Plan windows.

Adding a test library directory to the selections in a Device Plan window

Add a test-library selection as follows:

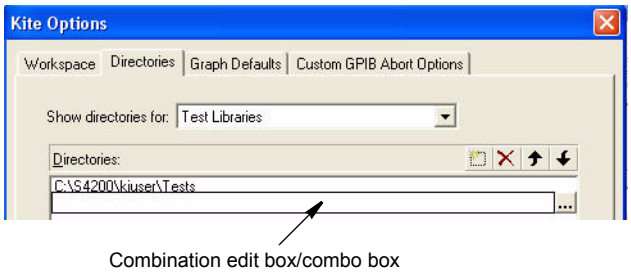
1. Open the **Directories** tab as described above under Customizing directory options later in this section.
2. In the **Directories** tab, in the **Show directories for** combo box, select **Test Libraries.** The **Directories** edit box displays the test library directories from which you can presently insert tests into a project plan.
3. In the **Directories** area of the **Directories** tab, click the **New** toolbar button, as shown Figure 6-405, or press the **INSERT** keyboard key.

Figure 6-405
"New" toolbar button

17. For example, the C:\S4200\kiuser\Tests factory-default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Tests.

A combination edit box/directory picker box now appears below the last displayed directory. See Figure 6-406.

Figure 6-406
Combination edit box/combo box for entering test library directory



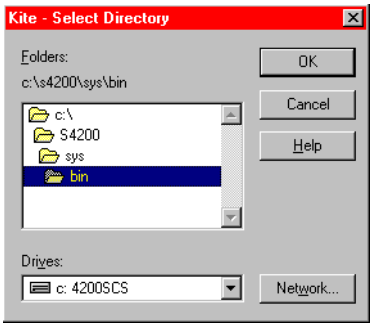
4. In the combination edit box/directory picker box that is displayed in the **Directories** area, enter the path and directory name of the test library to be added. Use one of the following methods:
   - **Method I**: Type in the path of the test library directory to be added. For example, see Figure 6-407.

Figure 6-407
Personal test directory name and path typed into the displayed edit box/combo box

   - **Method II**: Browse for the test library directory, as follows:

   a. Click the button at the right side of the edit box/directory picker box that is displayed in the **Directories** area. A KITE - Select Directory window appears. For example, see Figure 6-408.

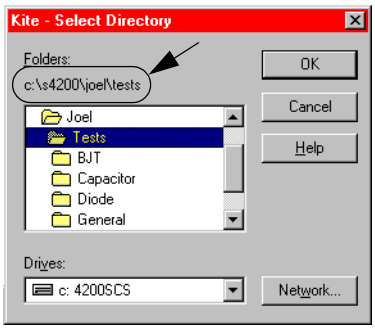
Figure 6-408
KITE - Select Directory window



   b. In the KITE - Select Directory window, browse for the directory to be added. If you need to map a network drive for a new directory, click the **Network** button and map the drive using the **Map Network Drive** window that appears.

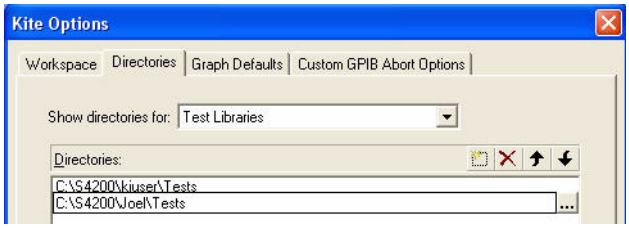
   c. Figure 6-409 shows the selection of a personal test library directory called `c:\S4200\Joel\Tests` using the browse method.

Figure 6-409
Selection of a personal test-library directory in a KITE - Select Directory window



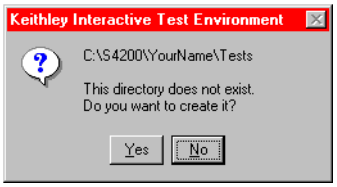
d. In the KITE - Select Directory window, click **OK**. KITE enters the browse selected test library directory into the test/combo box and then highlights the entry. See Figure 6-410.

Figure 6-410
Added browse selected test library directory



5. At the bottom of the **Directories** tab, click **OK**.
6. Complete this procedure according to one of the following:
   - If you just entered the path for an *existing* test library directory, this directory is now added to the list of directories from which you can insert project tests. Stop here. You have completed the procedure.
   - If you just entered the path for a *new* (not previously existing) directory, then KITE displays the message shown in Figure 6-411.

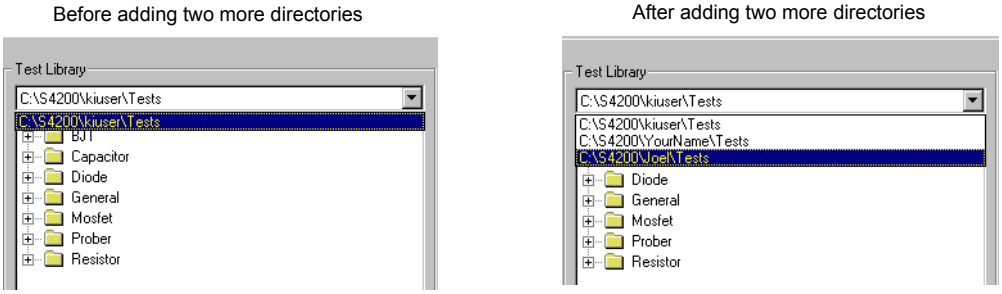
Figure 6-411
"Create new directory?" message box



- Click **Yes**. The following occurs:
  - The new, *empty* test directory is added to your Model 4200-SCS hard drive (or network drive, if selected).
  - The new directory is added to the list of directories from which you can insert project tests.

Figure 6-412 shows the **Test Library** combo box of a Device Plan window before and after adding two test-library directories.

Figure 6-412
Device Plan window before and after adding two test library directories



Changing the displayed position of a test library selection in the Device Plan window

If you use one test library more than others, it is convenient for this test library to be displayed by default in the **Test Library** combo box (when a Device Plan window first opens). To achieve this, reposition the frequently used directory at the top of the list in the **Directories** Tab of the KITE Options window. Do the following:

1. Open the **Directories** tab as described previously under Customizing directory options.
2. In the **Directories:** area of the **Directories** tab, select the test library directory to be repositioned.
3. Move the selected test library directory to the top of the list by clicking the **Move Up** or **Move Down** toolbar buttons (see below), or by pressing the **ALT** + **UP ARROW** and **ALT** + **DOWN ARROW** keyboard keys.

Figure 6-413
Move Up and Move Down toolbar buttons

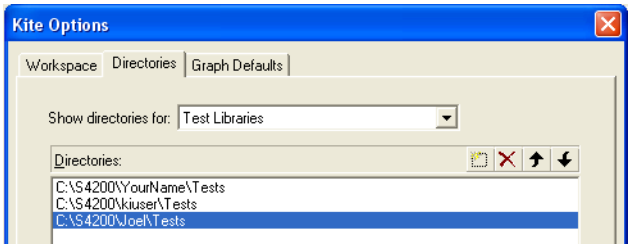
4. Click **OK**. The repositioned test library directory is now displayed by default when you open a Device Plan window.

Deleting a test library directory from the selections in the Device Plan window

To delete a test library directory from those displayed in a Device Plan window, do the following:

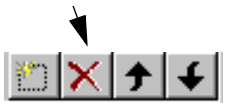
1. Open the **Directories** tab as described previously under Customizing directory options.
2. In the **Directories** area of the **Directories** tab, select the test library directory to be deleted. Figure 6-414 shows selection of the **Joel** test library directory.

Figure 6-414
Selection of the test library directory to be deleted



3. Delete the selected test library directory from the list by clicking the **Delete** toolbar button (see below), or by pressing the **DELETE** keyboard key. The selected test library disappears from the list.

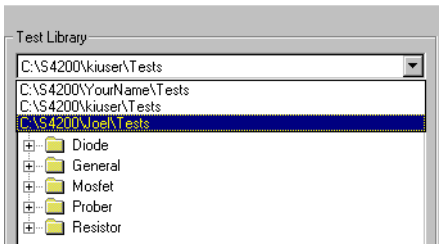
Figure 6-415
Delete toolbar button



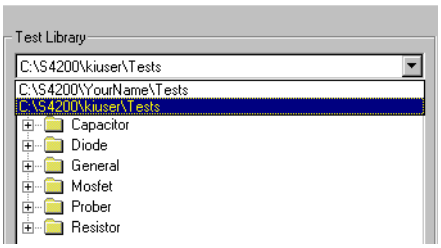
4. Click **OK**. The deleted test library directory is no longer displayed when you open a Device Plan window. See Figure 6-416.

Figure 6-416
Device Plan window before and after adding two test library directories

Before deleting a directory



After deleting a directory



Specifying which device library directories are to be available to projects

When choosing project devices through a Subsite Plan window or an Add New Device to Project window, you can normally choose these only from the device library that resides in the default user directory.[18] However, if you previously submitted devices to a library other than the default device library (as described in Submitting devices to a library), then you may desire to choose devices from multiple libraries. Therefore, KITE allows you to add (and delete) other device library selections to all Subsite windows and Add New Device to Project windows.

The procedures for adding, repositioning, and deleting a device library directory are essentially the same as described above, under Specifying which test library directories are to be available to projects, except as follows:

- In the **Directories:** area of the **Directories** tab, select the device library directory to be inserted instead of a test library directory.
- Replace all other uses of the word test with the word device.
- Replace use of the words Device Plan window with the words Subsite Plan window (and Add New Device to Project window).

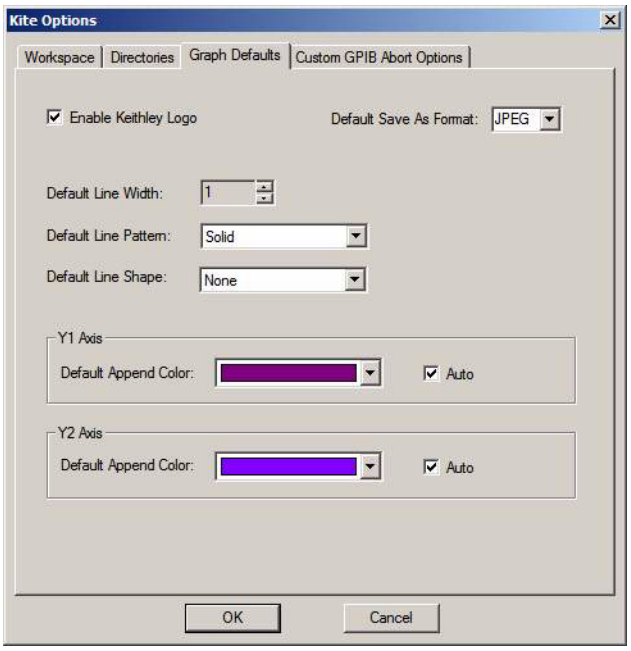
Customizing graph defaults

Figure 6-417 shows the KITE Options window to set the defaults for graphs:

- Enable or disable the Keithley logo in the graph.
- Set the file format for saving graphs (BMP, JPEG or TIFF).
- Set the line width, pattern and shape.
- Set the append colors for the X and Y axis.

18. For example, the C:\S4200\kiuser\Devices factory default directory or another directory that was specified as the default using KCON, such as C:\S4200\YourName\Devices.

Figure 6-417
Graph Defaults window

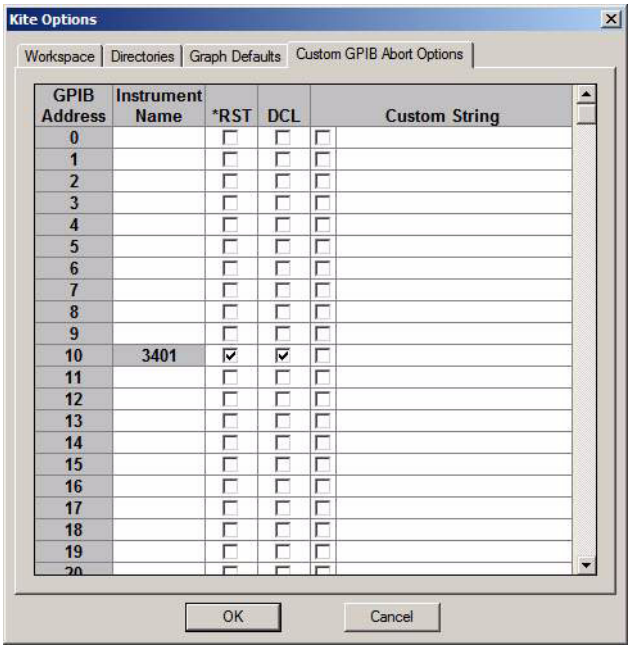


Custom GPIB Abort Options

Figure 6-418 shows the KITE Options window to set the operations that will occur when a GPIB abort is performed. In Figure 6-418, a *RST and DCL will be performed on the Keithley Instruments Model 3401 pulse generator when an abort occurs. Note that with Custom String enabled, a user-defined GPIB command string can be sent to the instrument.

NOTE *While most instruments will likely respond to the DCL command, instruments that are not SCPI compliant will not. Erratic operation may result. Refer to the instruction manual for the instrument to determine its capabilities.*

Figure 6-418
Custom GPIB Abort Options window



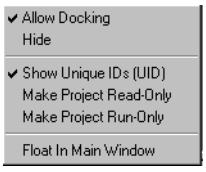
Customizing the view

Project Navigator display options

You can select whether or not KITE displays the Project Navigator, for example, when you desire a wider KITE workspace. Toggle the display of the Project Navigator by clicking **Project Navigator** in the **View** menu.

You can also specify the additional preferences for the Project Navigator through a pop-up menu, that is displayed when you right-click the project plan name (the first component in the project tree). See Figure 6-419.

Figure 6-419
Project Navigator pop-up menu



The pop-up menu selections are used as follows:

- **Allow Docking**: When checked, allows an undocked Project Navigator to be docked in the KITE window as follows:
  - On either side of the KITE window, by moving it to the extreme right or extreme left.
  - To the last docking position, by double-clicking its red **Project Navigator** window label (when docked, you *un*dock it by clicking the gripper bar at the same location).
- **Hide**: Clicking **Hide** hides the Project Navigator. To unhide it, select **Project Navigator** in the **View** menu.
- **Show Unique IDs (UID)**: When checked, the UIDs are displayed on the Project Navigator.

- **Make Project Read-Only**: If checked, the project and its data can be viewed, but the project cannot be modified or executed.
- **Make Project Run-Only**: If checked, the project can be viewed and executed. Its data can be viewed and saved, but the project cannot be modified. This setting allows operators to run tests and project sequences while protecting test and project definitions from accidental changes.
- **Float in Main Window**: When checked, causes the following:
  - Disables the **Allow Docking** selection.
  - Prevents the Project Navigator from docking, regardless of where it is moved in the KITE window.

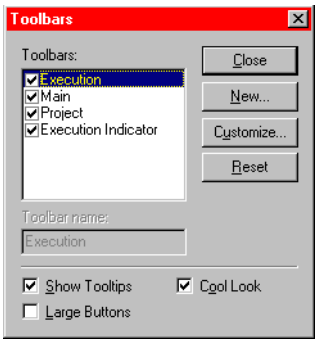
Messages display option

You can select whether or not KITE displays the **Messages** area at the bottom of the KITE window (for example, when you temporarily desire more height in the KITE workspace). Toggle the display of the **Messages** area by clicking **Project Messages** in the **View** menu.

Toolbar display options

You can select whether KITE displays the toolbars and toolbar buttons, and how it displays them, from the Toolbars window. To open the Toolbars window, click in the **View** menu on **Toolbars**. See Figure 6-420.

Figure 6-420
Toolbars menu



Selecting the toolbars to be displayed

In the Toolbars window under **Toolbars**, select the toolbars to be displayed by checking the appropriate checkboxes.

Selecting the toolbar/tool button style

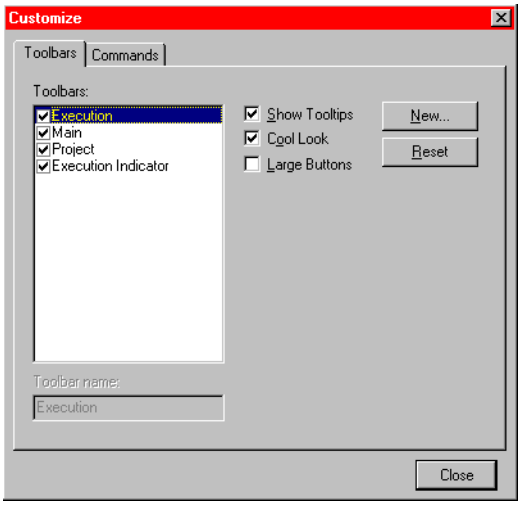
In the Toolbars menu, select toolbar/tool button styles using the three checkboxes, which are used as follows:

- **Show Tooltips**: When **Show Tooltips** is checked, KITE displays a brief description of a toolbar button when you place the cursor over the toolbar button.
- **Large Buttons**: When **Large Buttons** is checked, the toolbar buttons are about twice their normal size.
- **Cool Look**: When **Cool Look** is checked, the toolbar buttons have a gripper. When **Cool Look** is unchecked, the toolbar buttons have no gripper.

Customizing toolbars

In the Toolbars window, clicking the **Customize** button opens the Customize window. See Figure 6-421.

Figure 6-421
Customize window



The Customize window allows you to add toolbars and move and copy any of the existing tool buttons to them. Clicking the **Reset** button, here or on the Toolbars window, restores the default toolbars (but does not change any newly added toolbars).

Calibrating the system

NOTE *Before initiating a calibration, allow the system to warm up for at least 30 minutes after power-up.*

To maintain SMU performance specifications, you must initiate a Model 4200-SCS system auto-calibration every 24 hours or any time after the ambient temperature has changed more than ±1°C. Initiate an auto-calibration as follows:

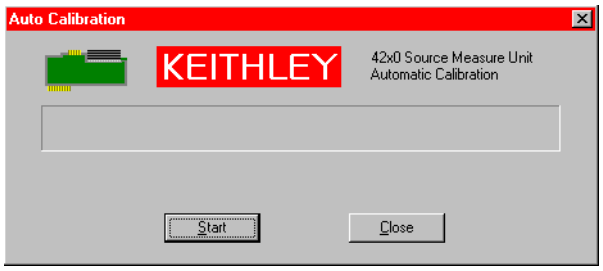
1. In the KITE Tools menu, click Auto Calibration. A disconnect-devices caution message appears. See Figure 6-422.

Figure 6-422
Disconnect devices caution message

2. Disconnect all devices from the Model 4200-SCS SMUs.

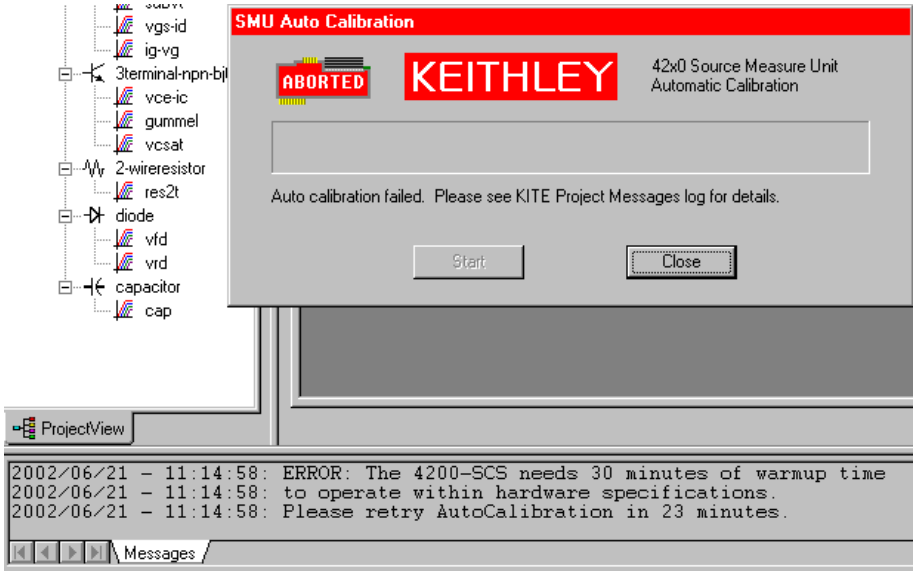
3. In the disconnect-devices caution box, click OK. The Auto Calibration dialog box opens. See Figure 6-423.

Figure 6-423
Auto Calibration dialog box



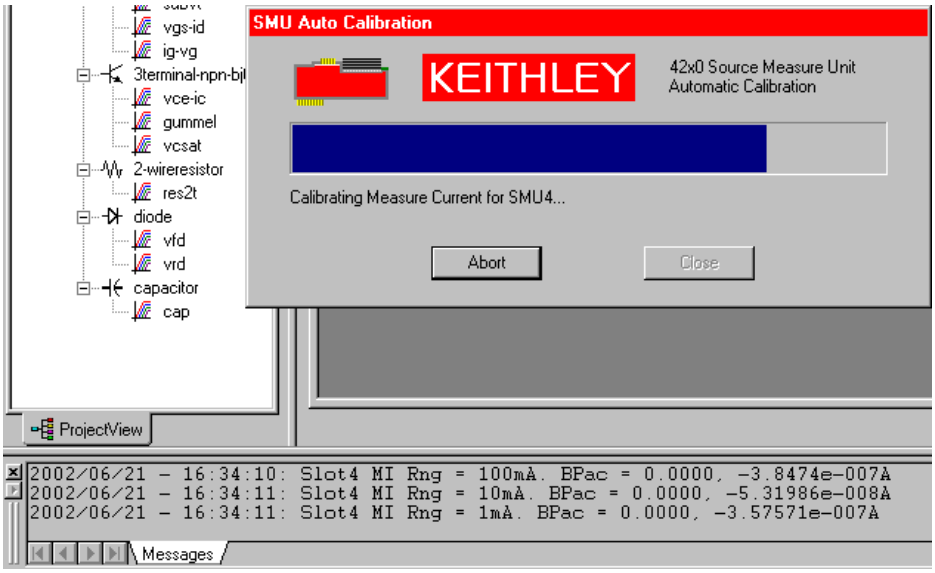
4. In the Auto Calibration dialog box, click **Start**.
   - If the system is insufficiently warmed up, KITE inhibits the auto calibration. Then, the Auto Calibration dialog box and the message area display messages similar to Figure 6-424.

Figure 6-424
Insufficient warm-up message



   - Otherwise, if the system is adequately warmed up, the calibration proceeds as follows:
     - The auto calibration routine recalibrates the current and voltage offsets for all source and measurement functions of all SMUs in the system.
     - The Auto Calibration dialog box and the message area display the progress of the auto calibration. For example, see Figure 6-425.

Figure 6-425
Progress display in the Auto Calibration dialog box and the message area



This page left blank intentionally.

5. When the auto-calibration is complete, the Auto Calibration dialog box displays the message shown in Figure 6-426.

Figure 6-426
Auto Calibration completion notification

6. In the Auto Calibration dialog box, click **Close**. The Auto Calibration dialog box closes.

Section 7

Keithley CONfiguration Utility (KCON)

In this section:

Introduction

The Keithley CONfiguration utility (KCON) is used to manage the configuration of the Keithley Instruments Model 4200-SCS and all external system components supported by the KTE Interactive software tools. Supported switch matrices, external GPIB instruments, and probe stations are added, configured, and removed from the system configuration using KCON. The KCON utility also provides basic diagnostic and troubleshooting functions.

If instrumentation is added to the system, or connections between the measurement instrumentation and the switch matrix are changed, KCON must be run to update the system configuration. Once the system is properly configured, Keithley Interactive Test Environment (KITE) and Keithley User Library Tool (KULT) can utilize the available resources in a configuration-independent manner.

KCON main window

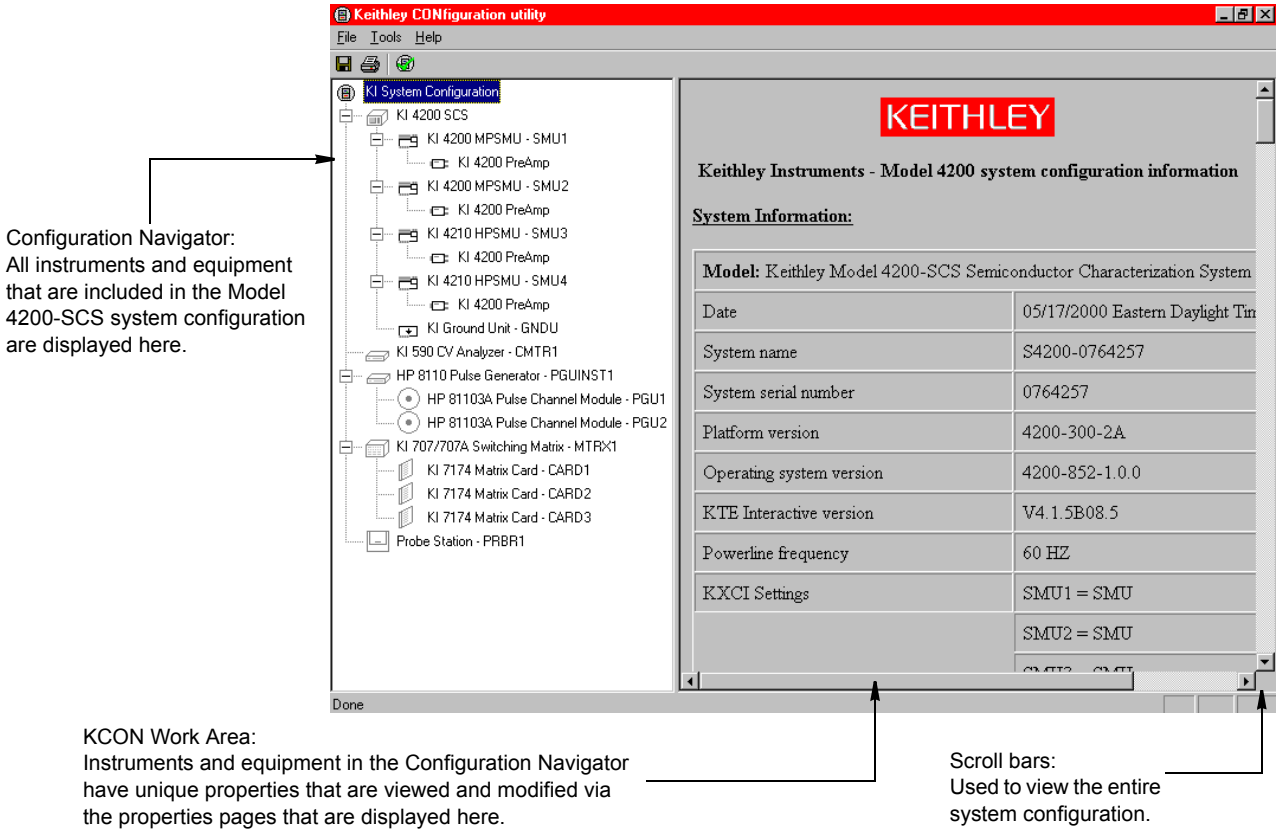
When KCON starts, the KCON main window (shown in Figure 7-1) appears. It has two panes; the left pane is the Configuration Navigator, and the right pane is the Workspace.

The Configuration Navigator provides a tree view of all instruments and equipment present in the Model 4200-SCS system configuration. The tree can be expanded and minimized by clicking on the plus (+) and minus (-) symbols, respectively.

The Workspace is a context-sensitive display area. Each instrument in the system configuration has associated properties. Selecting an instrument in the Configuration Navigator causes the associated-properties window to be displayed in the Workspace. Selecting a KI System Configuration node in the Configuration Navigator causes a summary of the entire system configuration to be displayed in the Workspace.

NOTE *Before starting KCON, ensure that KITE is not running. If KITE is running, the system configuration will be read-only. You cannot modify the system configuration while KITE is running.*

Figure 7-1
KCON main window



Configuration Navigator

The Configuration Navigator is a tree-type control containing each component present in the system configuration. Selecting a component, or node, in the Configuration Navigator causes the properties associated with the selected component to be displayed in the Workspace. In Figure 7-2, a typical system configuration is displayed with the Configuration Navigator completely expanded. To remove an external component from the system configuration, do one of the following:

- Select the component with the right mouse button, and then click a single-item. The **Remove** menu appears; select **Remove Prober**.

  or

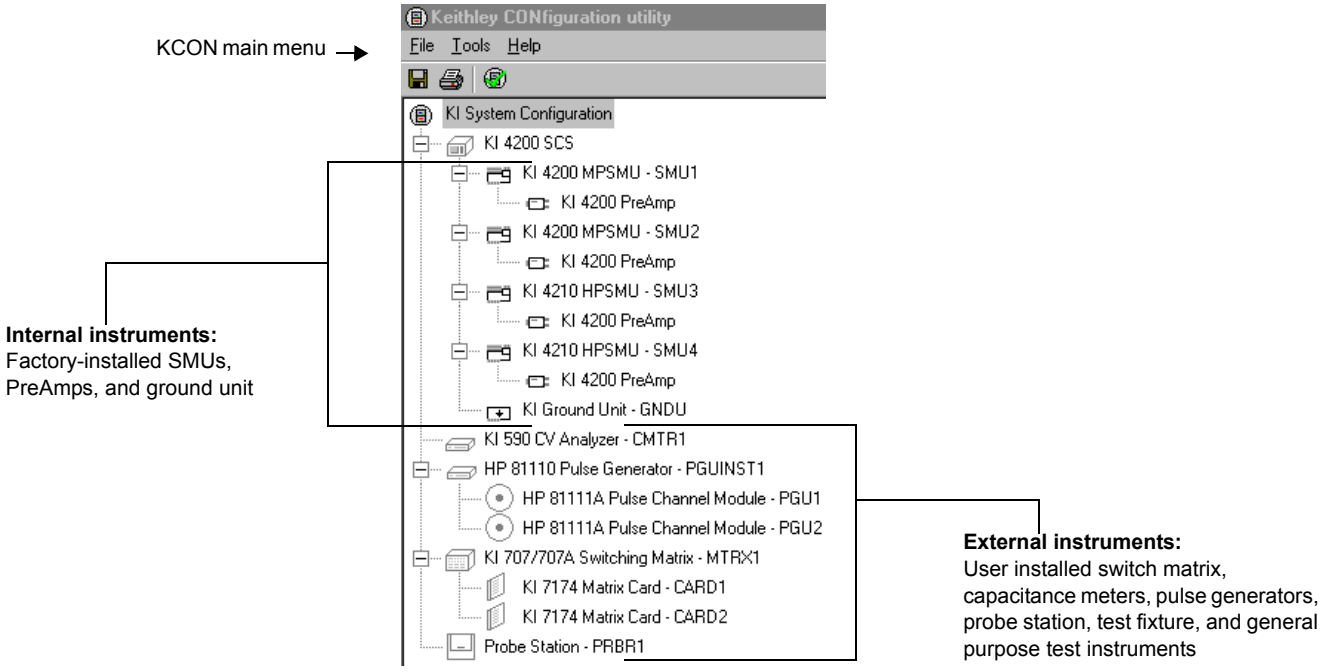
- Select the component and press the **Delete** key.

  or

- Select **Tools > Delete External Instrument**.

NOTE *Internal instruments cannot be deleted.*

Figure 7-2
Configuration Navigator view of the system configuration



KCON main menu

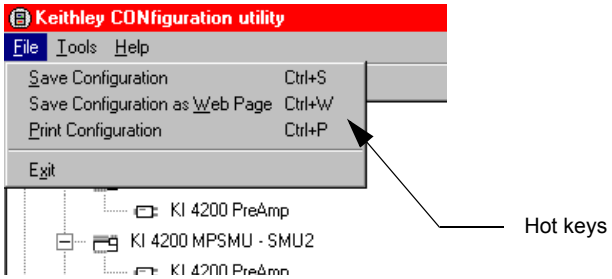
The KCON menu bar contains **File**, **Tools**, and **Help** pull-down menus:

- **File** Provides **Save Configuration**, **Save Configuration as a Web Page**, **Print,** and **Exit** functions. See Figure 7-3.
- **Tools** Provides selections to **Add External Instrument, Delete External Instrument**, **Validate Configuration, Update Preamp Configuration,** and access to a list of default **Formulator Constants.** This is illustrated in Figure 7-4.
- **Help** Provides selections to access the **Model 4200-SCS Complete Reference**, **Generate Technical Support Files**, and view version information **About KCON**. This is illustrated in Figure 7-8.

File menu

The KCON File menu is illustrated in Figure 7-3. Each menu item is described below.

Figure 7-3
File menu



File → Save Configuration

Save Configuration saves changes to the system configuration. If you do not save the configuration changes, KCON returns to the last saved configuration.

File → Save Configuration as Web Page

Save Configuration as Web Page saves the system configuration as a file in html format, so that it can be viewed with a web browser. If the **KI System Configuration** node in the Configuration Navigator is selected, **Save Configuration as Web Page** generates a web page that contains the information that is displayed in the KCON Workspace.

File > Print Configuration

Print Configuration prints the system configuration information (the information that is displayed in the KCON Workspace when the **KI System Configuration** node is selected in the Configuration Navigator).

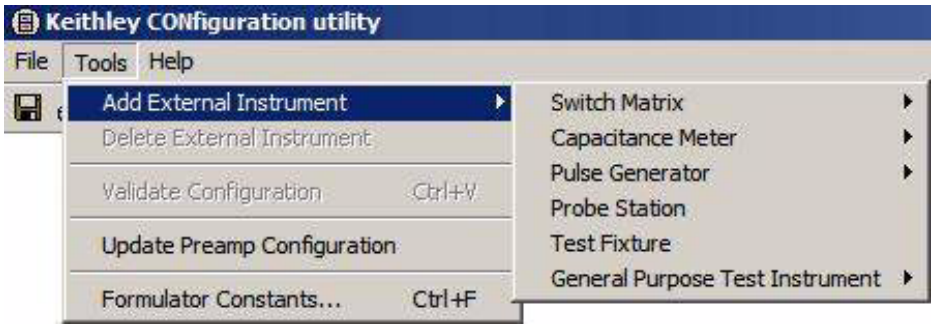
File > Exit

Exit closes the KCON program. If system configuration changes are pending, you are prompted to save them before exiting.

Tools Menu

Figure 7-4 shows the KCON **Tools** menu. Each menu item is described below.

Figure 7-4
Tools menu



Tools > Add External Instrument

Each supported external instrument is added to the system configuration by selecting it from the categorized **Add External Instrument** submenu, which is illustrated in Figure 7-4. The supported instrument categories are:

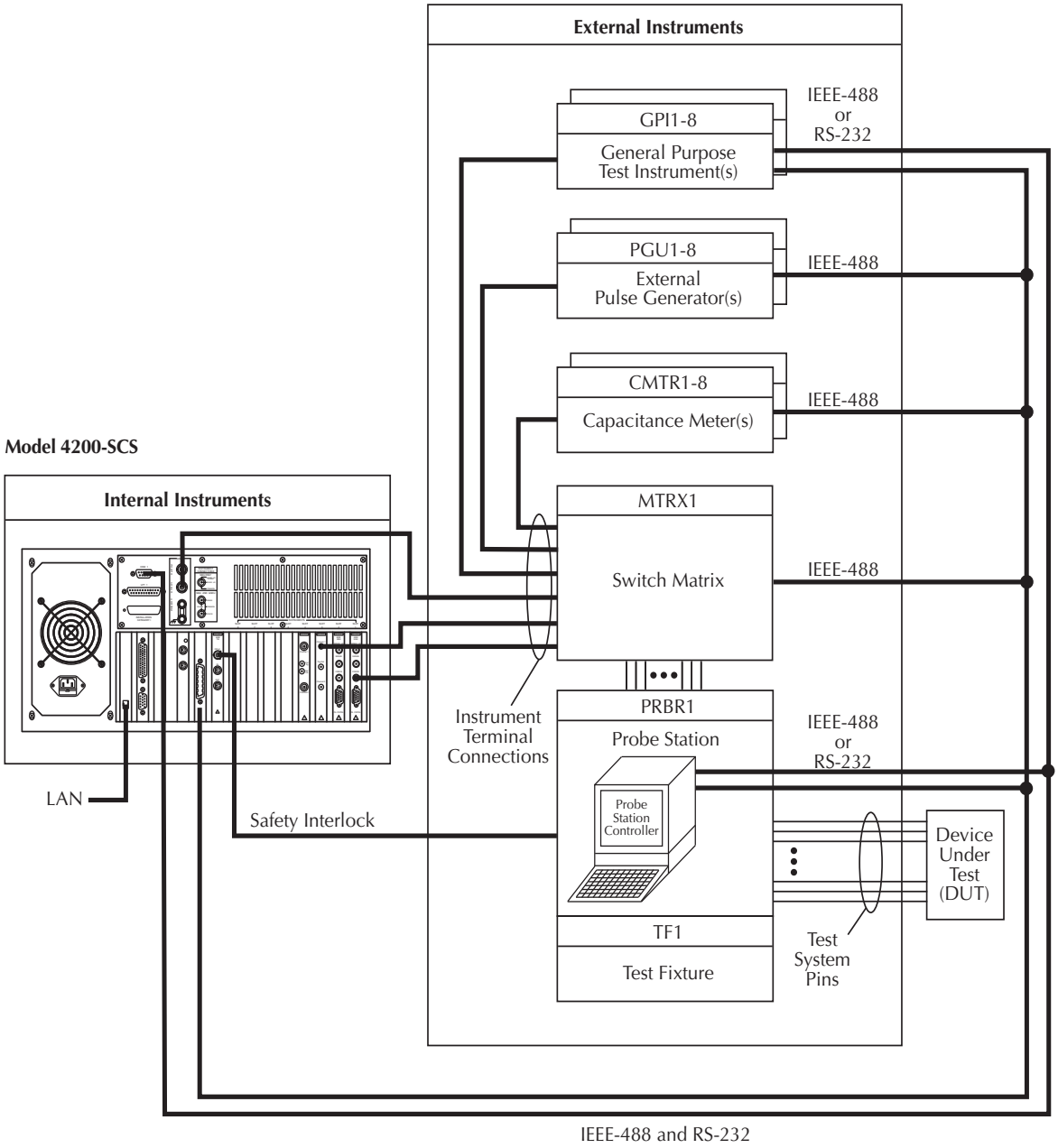
- Switch Matrix
- Capacitance Meter
- Pulse Generator
- Probe Station
- Test Fixture
- General Purpose Test Instrument

All supported external instrumentation and equipment is controlled by KITE User Test Modules (UTMs) that are connected to KULT user modules. Keithley Instruments provides libraries of user modules for each supported external instrument (refer to Table 7-1). Users can modify these libraries or create their own using the Keithley User Library Tool (KULT*)*. Additional information regarding external instrumentation and user modules can be found in the following locations:

- Configuring the UTMs in Section 6
- How to Create UTMs Using C Programming in Section 7 of the User's Manual
- Keithley User Library Tool (KULT) in Section 8

Figure 7-5 shows the relationship between internal and external instrumentation and illustrates each instrument category.

Figure 7-5
Typical configuration with external instruments



Keithley Instruments provides a number of standard user libraries to control external equipment that is typically used in semiconductor characterization applications. Standard user-module libraries are provided for the equipment shown in Table 7-1.

Table 7-1
Supported external equipment table

Category	Instrument	User library	Additional information
Switch matrix[1]	Keithley Instruments Model 707 / 707A Switching Matrix	matrixulib	"Using Switch Matrices" in Appendix B
	Keithley Instruments Model 708 / 708A Switching System		
Capacitance[2] meter	Keithley Instruments Model 590 CV Analyzer	ki590ulib	"Using a Keithley Instruments Model 590 CV Analyzer" in Appendix C
	Keithley Instruments Model 595 Quasistatic C-V Meter	ki595ulib	Model 595 Quasistatic CV Meter Instruction Manual (document number 595-901-01)
	Keithley Instruments Model 82 Simultaneous C-V System	ki82ulib	"Using a Keithley Model 82 C-V System" in Appendix E
	Agilent Model 4980 LCR Meter[6]	hp4980ulib	"Using an HP 4284A/4980A LCR Meter" in Appendix D
	Agilent Model 4980 LCR Meter[6]	hp4294ulib	"Additional User Libraries" in Appendix N.
Pulse generator[3]	Keithley Instruments Model 3401 Pulse Generator	ki340xulib	Model 3400 Series Pulse/Pattern Generators User's Manual (document number 3400S-900-01).
	Keithley Instruments Model 3402 Pulse Generator		
	Agilent Model 8110A Pulse Generator[6]	hp8110ulib	"Using an Agilent 8110A/81110A Pulse Generator" in Appendix J
	Agilent Model 81110A Pulse Generator[6]		
Probe station[4]	Suss MicroTec Model PA-200 semiautomatic probe station	prbgen	"Suss MicroTec PA-200 Prober" in Appendix H
	Micromanipulator Model 8860 semiautomatic probe station	prbgen	"Micromanipulator 8860 Prober" in Appendix I
	Manual probe station and fake probe station	prbgen	"Using a Manual or Fake Prober" in Appendix J
	Cascade Summit-12000 probe station	prbgen	"Cascade Summit-12000 Prober" in Appendix K
	Signatone CM500 Prober	prbgen	"Signatone CM500 Prober" in Appendix L
Test fixture	Keithley Instruments Model 8006 Component Test Fixture	Not applicable	"Connections and Configuration" in Section 4
	Keithley Instruments Model 8007 Semiconductor Test Fixture		
	Generic test fixture		
General purpose test Instruments[5]	Any IEEE-488 controlled or RS-232 controlled instrument or equipment	Created by user	

1. The Model 4200-SCS supports the Keithley Instruments Model 707/707A and 708/708A Switch Matrices. The Model 708/708A accepts a single matrix card. The Model 707/707A accepts up to six matrix cards. Only one switch matrix can be present in the system configuration at a time.
2. Up to eight supported capacitance meters may be added to system configuration.
3. The Model 4200-SCS supports a maximum of eight external pulse generators and a maximum of 16 pulse generator unit (PGU) channels. In other words, eight dual-channel pulse generators can be present in the system configuration at one time, providing a total of 16 PGU channels. If eight single-channel pulse generators are used, the number of available PGU channels is eight. Combinations of single and dual-channel pulse generators can be used. For example, if three dual-channel and five single-channel pulse generators are used, the total number of PGUs is 11. For limits on maximum number of internal pulse generators in a test (such as the Model 4220-PGU or Model 4225-PMU), see "Model 4200-SCS power supply limitations" on page 16-73.
4. For general information about using a probe station, refer to "Using a Probe Station" in Appendix G.
5. The Model 4200-SCS supports up to eight general purpose test instruments (GPIs). two-terminal and four-terminal types may be present in the system configuration simultaneously, but the total number of GPIs cannot exceed eight.
6. HP and Agilent are used interchangeably based on menu preference and written description.

Tools > Delete External Instrument

Delete External Instrument removes an external instrument from the system configuration.

NOTE *You can remove an external instrument by selecting it in the Configuration Navigator and then pressing the right mouse button. This action displays a single-item pop-up menu; clicking the pop-up menu item deletes the instrument. Alternatively, you can delete an external instrument by selecting it and then pressing the* **DELETE** *keyboard key.*

Tools > Update DC Preamp and RPM Configuration

Update preamp configuration:

A preamp can be physically removed or reconnected to the SMU while the system is running. However, after adding or removing the preamp, the system must be updated by clicking the **Update Preamp and RPM Configuration** option in the **Tools** menu. This update must also be performed if the Model 4200-SCS was turned off when the preamp was removed or added. The system will not automatically update during power-on.

Update RPM configuration:

1. Turn off system power before connecting or disconnecting the RPM to the PMU.
2. With the system off, connect the correct RPM to the correct PMU channel.
3. Connect any SMU to any RPM using either one or two triax cables. Repeat for all RPMs in the system. If you have no SMUs, skip this step.
4. Connect CVU High to one RPM, and CVU Low to the other RPM. If you have no CVU, skip this step.
5. Turn on the system power and start KCON.
6. Select **Tools** > **Update PreAmp and RPM Configuration**.
7. Select **File** > **Save** before exiting KCON.

When using a KITE ITM for any DUT testing, the RPM will automatically set to select either the PMU, SMU, or CVU. However, for UTM testing, UTMs must be used to select the correct RPM settings. See Controlling RPM switching in Section 16 for more information.

Tools > Validate Configuration

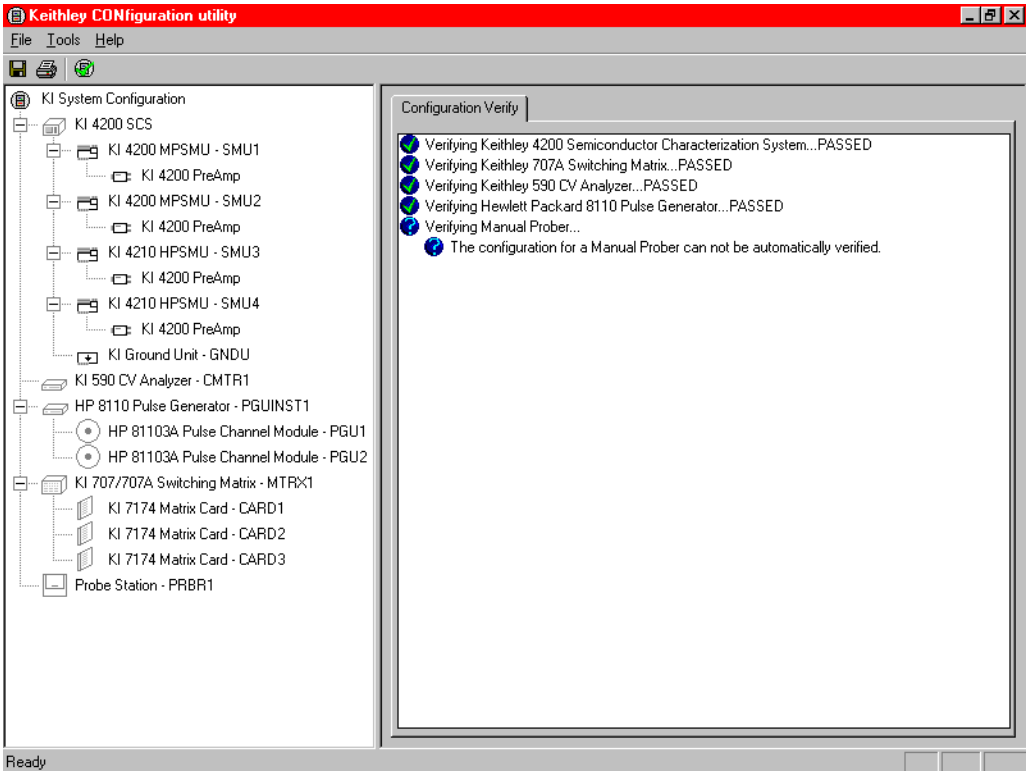
Validate Configuration tests the system configuration to determine if there are configuration conflicts or communication problems between the instrumentation and the Model 4200-SCS. Most of the supported internal and external instruments can be validated. To be more specific, when **Validate Configuration** executes, KCON communicates with the instruments to verify that the physical configuration matches the KCON defined configuration. However, instruments in the following categories are not automatically verified when the configuration is validated:

- Probe stations
- Test fixtures
- General purpose test instruments

Figure 7-6 shows a sample of the report that KCON generates when it validates the system configuration (as defined in the main window in Figure 7-1). When KCON detects configuration conflicts, it displays corresponding error messages in the Workspace.

NOTE *KITE automatically validates the configuration when it starts up. If KITE detects conflicts, it displays an error message instructing you to resolve the conflicts using KCON.*

Figure 7-6
Typical Validate Configuration report

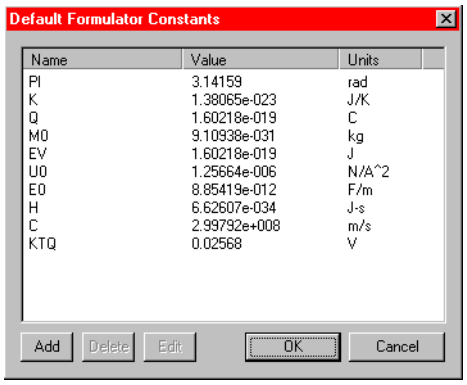


Tools > Formulator Constants

Formulator Constants modifies the default Formulator constants that are automatically assigned to new KITE test modules when they are created.

Measurement data can be manipulated by KITE test modules with a parameter extraction tool called Formulator. Formulator allows you to perform simple and complex data calculations on test data, as well as on other Formulator calculations. Formulator provides a variety of calculation functions, common mathematical operators, and common physical constants. Figure 7-7 below shows the standard physical constants that are provided by Keithley Instruments. The default Formulator constants can be modified using the **Add**, **Delete**, and **Edit** buttons.

Figure 7-7
Factory "default" Formulator constants

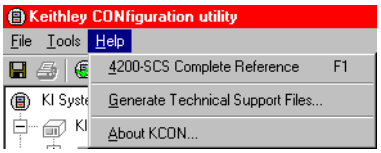


- PI (π) is the ratio of the circumference to the diameter of a circle.
- K is Bolzmann's constant.
- Q is the charge of an electron.
- MO is the electron mass.
- EV is electron volt.
- UO is permeability.
- EO is the permittivity of a vacuum.
- H is Planck's constant.
- C is the speed of light.
- KTQ is the thermal voltage.

Help menu

The **Help** menu is illustrated in Figure 7-8. Menu items are discussed individually below.

Figure 7-8
Help pull-down menu



Help → Model 4200-SCS Complete Reference

4200-SCS Complete Reference loads the Model 4200-SCS Complete Reference portable website, which is preinstalled on your Model 4200-SCS and included on CD-ROM. It was specifically designed to provide easy access to all Model 4200-SCS reference information, such as:

- **Product manuals** The Model 4200-SCS User's and Reference Manuals, and related product manuals in searchable .pdf format
- **Data sheets** The Model 4200-SCS Technical Data Sheet and related product data sheets
- **Application notes** Pragmatic examples of how to use the Model 4200-SCS and related products, to perform application-specific tasks

Selecting **Help → 4200-SCS Complete Reference** automatically starts the web browser and loads the Complete Reference website.

Help → Generate Technical Support Files

Generate Technical Support Files performs a detailed analysis of your Model 4200-SCS, after prompting you for some contact information. KCON stores the analysis results on a floppy disk. The results can then be sent to Keithley Instruments for review.

To generate a technical support file:

1. Select **Help → Generate Technical Support Files.** KCON prompts you to insert a floppy disk. (KCON automatically copies the technical support files to this disk during the analysis).
2. Insert the floppy disk.
3. Click **Yes**.

4. After the System Audit window appears and indicates a successful analysis, do the following:
   a. Click **OK**. The System Audit window closes.
   b. Remove the floppy disk, which now contains technical support files.
   c. Send the floppy disk to Keithley Instruments. Contact your local Keithley Instruments sales office to set up the return of the floppy disk to the factory.

Technical support personnel at Keithley Instruments will review the analysis information and quickly assess the state of your Model 4200-SCS. This will enable them to efficiently resolve system problems.

Help → About KCON

About KCON displays a window that contains version and copyright information. See Figure 7-9 below.

Figure 7-9
The About KCON window

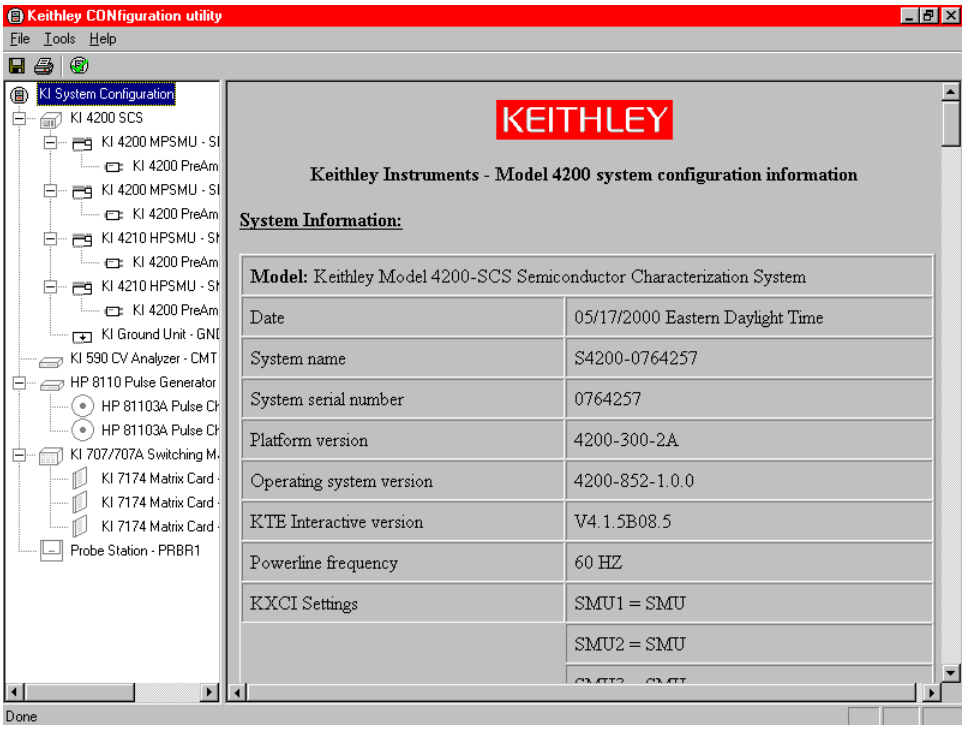
System Configuration properties

Each instrument that is included in the system configuration appears in the Configuration Navigator. Selecting a Configuration Navigator node causes the corresponding attributes or properties to be displayed in the Workspace. The following subsections describe the properties for each supported instrument.

KI System Configuration Properties window

When you select **KI System Configuration** in the Configuration Navigator, the Workspace displays the system configuration. See Figure 7-10 below.

Figure 7-10
KI System Configuration information



NOTE *The system configuration typically spans multiple pages. Use the scroll bar to view the complete record.*

The three sections of the system configuration are:

System Information Provides a table containing the Model 4200-SCS serial number, the network or system name, and pertinent software and platform version information.

Instrumentation Provides a table of properties and settings for each instrument in the system configuration.

Connections Provides a table of system connections. The connection table provides guidance when making connections between the instrumentation and an optional switch matrix and test fixture or probe station.

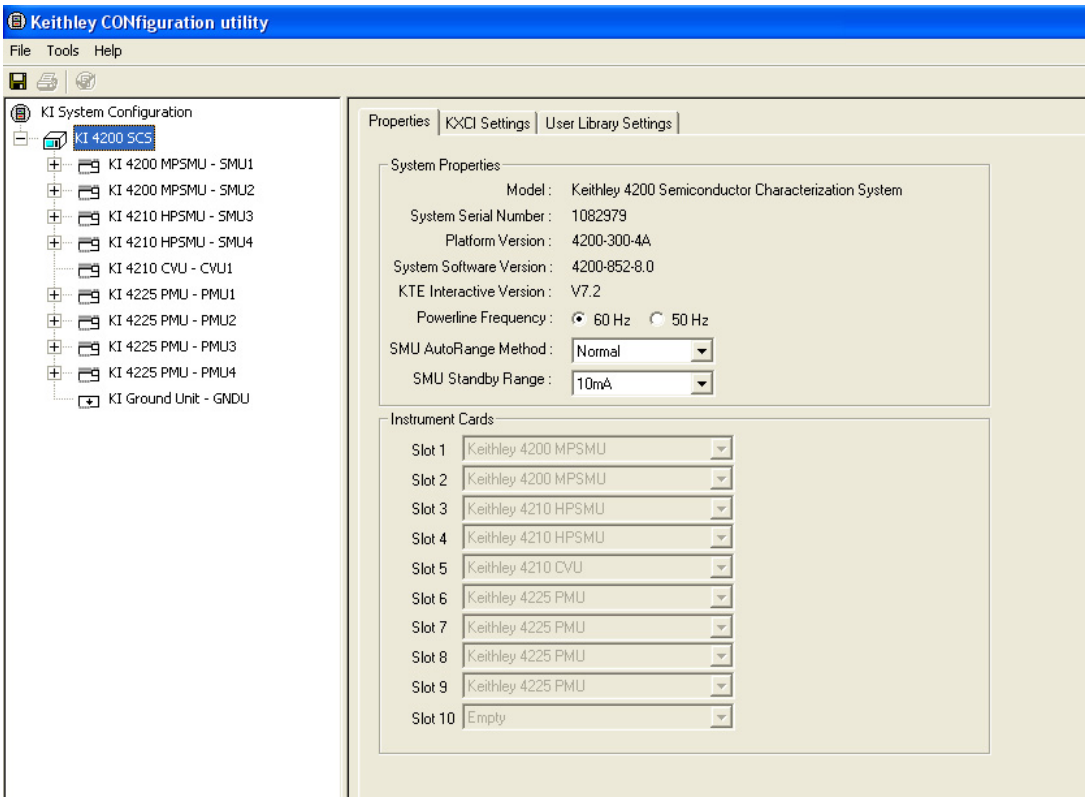
KI 4200 SCS Properties window

Selecting **KI 4200 SCS** in the Configuration Navigator causes the Model 4200-SCS system properties to be displayed in the Workspace. These properties are located on three tabs): **Properties**, **KXCI Settings**, and **User Library Settings**. Each tab is discussed below.

Properties tab

When you select the **KI 4200 SCS** node, the tab that first displays is the **Properties** tab. See Figure 7-11 below.

Figure 7-11
KI 4200 SCS Properties tab



The two areas of this **Properties** tab are described below.

System Properties area

The **System Properties** area of this **Properties** tab displays the Model 4200-SCS serial number and other pertinent software and platform information. System properties that you can change include the Power-line frequency, SMU autorange method, and SMU standby range.

Power-line frequency

Set **Powerline Frequency** to **60 Hz** or **50 Hz**. Make sure you set the power-line frequency correctly. If the setting is incorrect, the Model 4200-SCS cannot properly reject power-line related measurement noise.

SMU autorange method

The precision DC SMUs installed in the Keithley 4200-SCS are capable of making a smart decision on range changing. There are a variety of factors the SMU considers when making a range change decision. The SMU autorange method allows you to tune the range change algorithm to meet your particular need. There are three available SMU autorange methods; normal, high-speed, or high-resolution.

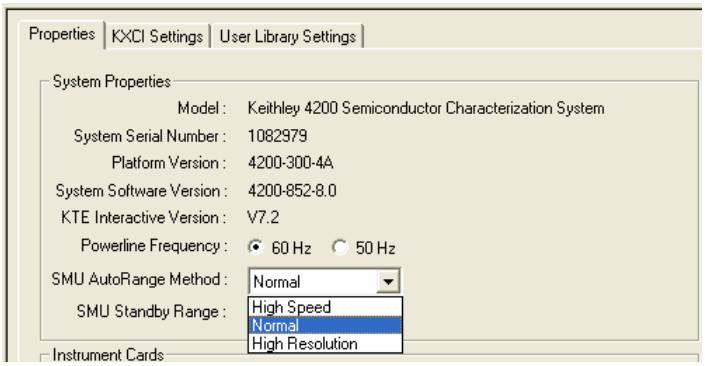
For the normal autorange method, the SMU makes a range decision at 50 percent of range. This allows for a relatively high precision measurement, with a minimum of range disruptions to the measurement. This mode works reliably in most conditions, and should be used unless there is a precision, speed, or glitch condition that you are attempting to correct.

For the high-speed autorange method, the SMU ranges at 10 percent of range. This approximately doubles the speed of the measurement. The SMU is capable of 6-digit precision, but a digit of precision is lost when using the high-speed autorange method. However, 5-digit precision is often enough for most applications. The high-speed autorange method is also the best choice if the device or test configuration tends to exhibit unstable characteristics, such as oscillations or unstable device readings.

For the high-resolution autorange method, the SMU ranges at 100 percent of range. This is the slowest autorange method, but allows full-precision of the SMU to be utilized. This also the lowest noise autorange method. Occasionally, this method causes a small measurement glitch on a complete sweep. In that case, changing to the high-speed autorange method will often resolve this problem.

Set the **SMU AutoRange Method** using the drop-down menu shown in Figure 7-12. Select the **High Speed**, **Normal**, or **High Resolution** autorange method.

Figure 7-12
SMU autorange method menu



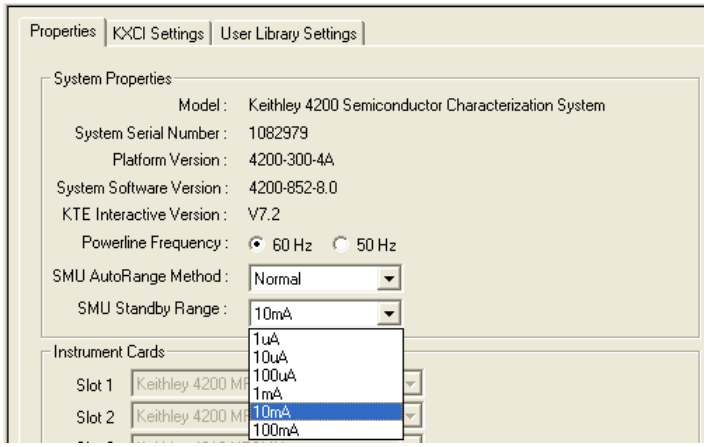
SMU standby range

When the SMU is NOT performing a test, it is in standby mode. When in standby, the SMU is programmed to output zero volts, with current compliance set to the maximum on the standby current range you selected. This allows the SMU to look like a virtual short circuit, and to prevent unwanted static charges from building up. The default standby range for the SMU is 10 mA.

The main reason to select a standby range (other than the default), is to tune or correct a test condition or result that is less than optimal. For example, if the test condition always starts the measurement on the 100 µA range, selecting the 100 µA standby range speeds up the measurement. This may also correct the occasional data glitch that is sometimes seen on the first point of a sweep.

Set the **SMU Standby Range** using the drop-down menu shown in Figure 7-13.

Figure 7-13
SMU standby range menu



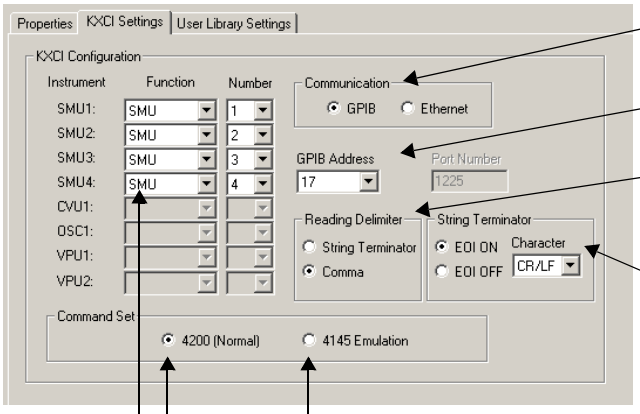
Instrument Cards area

The **Instrument Cards** area of this **Properties** tab shows which internal instrument card is installed in each slot.

KXCI Settings tab

The **KXCI Settings** tab stores the Keithley External Control Interface (KXCI) settings. KXCI allows the Model 4200-SCS to behave as a GPIB slave. To access the **KXCI Settings** tab (see Figure 7-14), select **KI 4200 SCS** in the Configuration Navigator.

Figure 7-14
KI 4200 SCS KXCI Settings tab



Select the communications interface (**GPIB** or **Ethernet**)

Set the **GPIB Address** or the **Port Number** for the Ethernet

For the GPIB, specifies whether the **Delimiter** for output data is a **String Terminator** or a **Comma**

For the GPIB, an **EOI** (End Or Identify) signal is sent after data transfer when this radio button is **ON,** and is not sent if it is **OFF**.

Select alternative SMU functions, in any mix, via these combo boxes.

Select the Keithley Instruments 4200 extended mode (the default mode) or the Agilent 4145 emulation mode. For more information about the two modes, refer to the description in the text below.

KXCI facilitates using an external computer to remotely control the Model 4200-SCS over the GPIB bus and Ethernet. In many cases, test programs developed for use with an Agilent 4145B run without modification when they are used with a Model 4200-SCS running KXCI. Refer to Keithley External Control Interface (KXCI) in Section 9 for detailed information regarding KXCI.

The **Function**, **Communication**, **GPIB Address/Port Number**, **Delimiter**, **EOI**, and **Command Set** settings on the **KXCI Settings** tab are each described in the following paragraphs.

Function combo boxes

Although the KXCI GPIB command set is very similar to the Agilent 4145B GPIB command set, the Model 4200-SCS and Agilent 4145B hardware are substantially different. The fundamental difference is that the Model 4200-SCS hardware is modular, whereas the Agilent 4145B hardware is fixed (refer to Table 7-2).

Table 7-2
Hardware comparisons

Instrument type	Keithley Instruments Model 4200-SCS	Agilent 4145B
Source measure units (SMUs)	2 to 8	4 (fixed)
Voltage monitor (VM)	Any SMU can be configured to function as a VM. Up to 8 VMs are possible.	2 (fixed)
Voltage source (VS)	Any SMU can be configured to function as a VS. Up to 8 VSs are possible.	2 (fixed)

KCON manages these hardware differences by allowing you to assign VM or VS functions to any Model 4200-SCS SMUs, using the **Function** combo boxes. Refer to Table 7-3.

Table 7-3
KXCI SMU (Source measure unit) function assignment

Function combo box selection	Description
SMU (source measure unit)	Instructs the Model 42XX-SMU to emulate the capabilities of an Agilent 4145B Source Measure Unit.
VM1...VM8 (voltage monitor)	Instructs the Model 42XX-SMU to emulate the capabilities of HP 4145B VM1 or VM2, as well as additional voltage monitors (VMs). Up to eight VMs may be assigned (mapped) to SMUs, depending on the number of SMUs in the system. A VM may be assigned any number from 1 to 8, regardless of the number of SMUs in the system. However, each VM number must be unique; duplicate VM numbers are not allowed.
VS1...VS8 (voltage source)	Instructs the Model 42XX-SMU to emulate the capabilities of HP 4145B VS1 or VS2, respectively, as well as additional voltage sources (VSs). Up to eight VSs may be assigned (mapped) to SMUs, depending on the number of SMUs in the system. A VS may be assigned any number from 1 to 8, regardless of the number of SMUs in the system. However, each VS number must be unique; duplicate VS numbers are not allowed.

NOTE *An invalid function selection defaults to the SMU function.*

Communications radio buttons

Use to select the communications interface (**GPIB** or **Ethernet**).

GPIB Address combo box

With GPIB communication selected, use the **GPIB Address** to select the primary address of the Model 4200-SCS when operating under KXCI control. This field is inactive when **Ethernet** is selected.

NOTE *GPIB address 31 is reserved for the Model 4200-SCS when it is operating as a system controller.*

If the selected GPIB address conflicts with the GPIB address of another system component, a red exclamation-point symbol (!) is displayed next to the selected address.

Port Number combo box

With Ethernet communication selected, set the **Port Number**. This field is inactive when the **GPIB** is selected.

Delimiter radio buttons

With GPIB communication selected, the output data delimiter character(s) are added to the end of each KXCI output message. Select **CR/LF** to configure KXCI to terminate output data with a carriage return and line feed character sequence. Select **Comma** to terminate output data with a comma (,). This field is inactive when **Ethernet** is selected.

EOI radio buttons

With GPIB communication selected, the **EOI** setting determines whether or not the Model 4200-SCS asserts the IEEE-488 End Or Identify (EOI) signal with the last byte of each output data message. Select **ON** to enable assertion of the EOI signal and **OFF** to disable the EOI signal. This field is inactive when **Ethernet** is selected.

Command Set radio buttons

By selecting one of the following **Command Set** buttons, you choose the control mode through which KXCI runs the Model 4200-SCS:

- **4200 (Normal)** Selects the 4200 extended mode. In this mode (a superset of the 4145 emulation mode (see below)), the Model 4200-SCS provides the full range of SMU capabilities. The SMU capabilities under the 4200 extended mode are essentially the same as under local control using the Model 4200-SCS display, keyboard, and pointing device.
- **4145 Emulation** Selects the 4145 emulation mode. In this mode, the KXCI GPIB command set, status model, and output data, are similar in function and format to those supported by the Agilent 4145B Semiconductor Parameter Analyzer. The capabilities of this mode (relative to the 4200 extended mode) more closely approximates the capabilities of the 4145B, though with substantially expanded performance.

Table 7-4 summarizes some differences and similarities between the two modes.

Table 7-4
Mode comparisons

Characteristic	Mode support	
	4145 emulation mode	**4200 extended mode**
String reported in response to ID query	ID HP4145B 1.1,1.0	KI4200 Vx.x.x (where x.x.x is the present version number)
GPIB data resolution	5 digits	7 digits
Maximum number of sweep data points	1024	4096
Possible instrument configurations	8SMU/VM/VS	
Configuration query	Not supported	*OPT? command

Table 7-4
Mode comparisons (continued)

Characteristic	Mode support	
	4145 emulation mode	**4200 extended mode**
Instrument self test	Not supported	SMUs only
Custom A/D control	Not supported	IT4 command options
200 V, 1A capability	Supported	
1.0 pA source/measure-range capability (with preamp on SMU)	Supported	

KXCI always starts in the selected mode, unless you change it using the **4200 (Normal)** or **4145 Emulation** button.

For additional details about the differences between the 4200 extended mode and 4145 emulation mode, refer to Keithley External Control Interface (KXCI) in Section 9.

NOTE *In many cases, test programs developed for use with an Agilent 4145B run without modification when they are used with a Model 4200-SCS running KXCI. Refer to "Keithley External Control Interface (KXCI)" in Section 9 for detailed information regarding KXCI.*

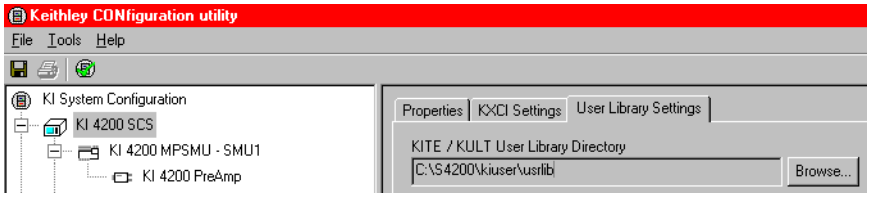
User Library Settings tab

The **Library Settings** tab allows you to specify the active KITE / KULT user library directory. The user library directory specified in KCON is used by KITE when user libraries are executed, and by KULT when user libraries are modified.

To set the active user library directory, do the following:

1. Select **KI 4200 SCS** in the Configuration Navigator.
2. Select the **User Library Settings** tab in the Workspace. See Figure 7-15.

Figure 7-15
User Library Settings tab

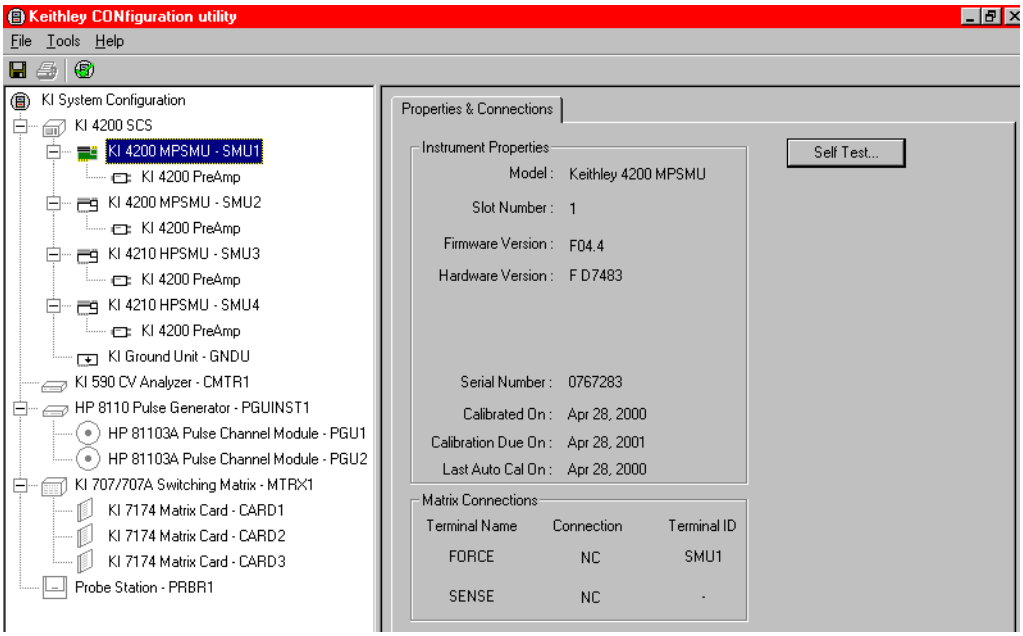


3. Click on the **Browse** button. The **Browse for Folder** window appears with the active user library directory highlighted.
4. In the **Browse for Folder** window, change the active directory by selecting another directory.

KI 4200/4210 SMU Properties and Connections tabs

Selection of a **KI 4200 MPSMU** in the Configuration Navigator causes the corresponding KI 4200 MPSMU **Properties & Connections** tab to be displayed in the Workspace. Additionally, selection of a **KI 4210 HPSMU** in the Configuration Navigator causes the corresponding KI 4210 HPSMU **Properties & Connections** tab to be displayed. A KI 4200 MPSMU **Properties & Connections** tab is shown in Figure 7-16.

Figure 7-16
KI 4200 MPSMU Properties & Connections tab



This **Properties & Connections** tab is divided as follows:

- **Instrument Properties** area — Provides useful hardware and software version information.
- **Matrix Connections** area — Provides switch-matrix connection information.
- **Self Test** button — Accesses the built-in SMU **Self Test** diagnostic utility.

These three areas are described below.

Instrument Properties area

The **Instrument Properties** area provides a variety of useful hardware and software version information, as well as calibration information.

Matrix Connections area

The **Matrix Connections** area displays the matrix connections that are associated with the SMU measurement terminals when if the following conditions apply:

- A matrix is included in the system configuration.
- The SMU is connected to the matrix.

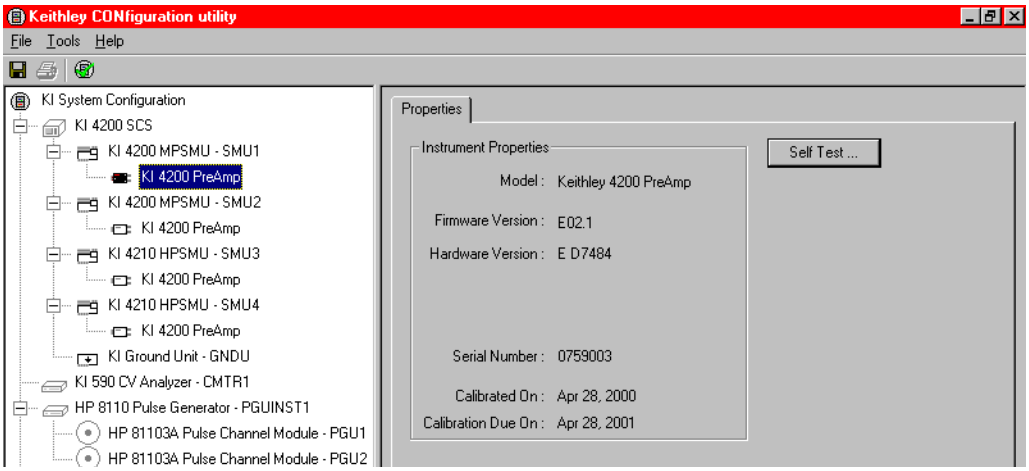
Self Test button

The **Self Test** button starts the **Self Test** utility, which aids in troubleshooting potential SMU hardware problems. When you run **Self Test,** the SMU performs several internal performance checks to determine if it is operating correctly. No external equipment is required. When the **Self Test** process finishes, a pass/fail window appears.

KI 4200 PreAmp Properties tab

When you select a **KI 4200 PreAmp** node in the Configuration Navigator, a **Properties** tab appears in the Workspace. See Figure 7-17.

Figure 7-17
KI 4200 PreAmp Properties tab



Instrument Properties area

The **Instrument Properties** area of the preamp **Properties** tab contains useful hardware and software version information, as well as calibration information.

PreAmp Self Tests button

The **Self Test** button of the preamp **Properties** tab starts the **Self Test** utility, which aids in troubleshooting potential preamp hardware problems. When you run **Self Test,** the preamp performs a number of internal performance checks to determine whether the preamp is operating correctly. However, in contrast to the SMU self test, the preamp self test requires the user to perform the following physical operations, *when prompted during the* **Self Test** *process:*

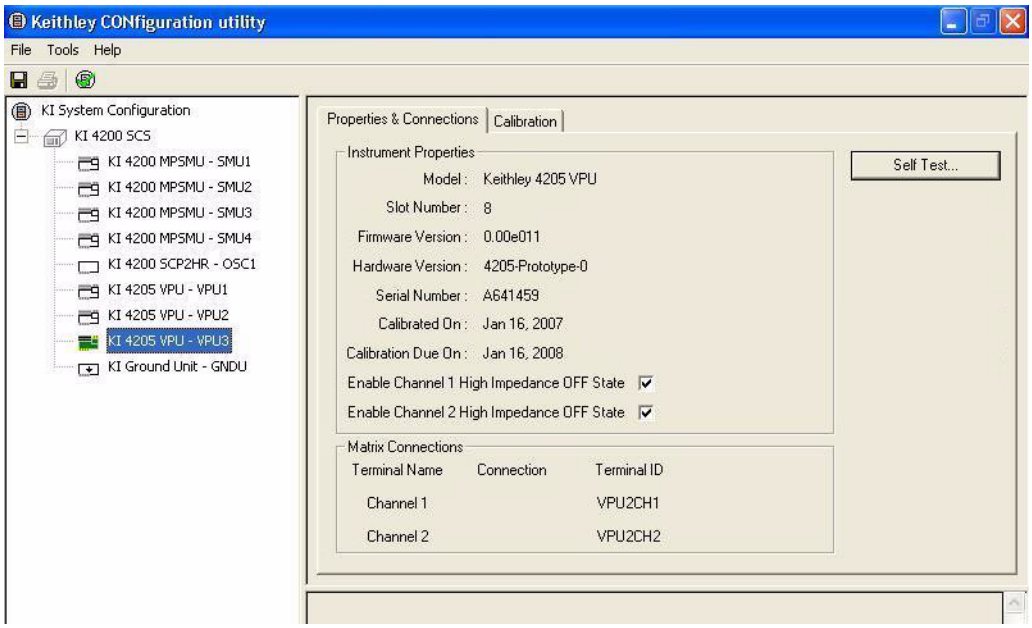
- Disconnect all cables from the preamp terminals, so that a sequence of open circuit tests can be run.
- Connect the preamp to the Ground Unit, so that short circuit tests can be run.

KI 4205 VPU Properties and Connections tab

When you select a **KI 4205 VPU**[1] in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-18.

1. The Models 4205-PG2, 4220-PGU, and 4225-PMU are dual-channel pulse cards inside the Model 4200-SCS. To differentiate between a Keithley pulse card and other supported pulse instruments, the Keithley pulse card may be referred to as a VPU or Voltage Pulse Unit. In KCON, a Model 4205-PG2 is referred to as a KI 4200 VPU.

Figure 7-18
KI 4205 VPU Properties & Connections tab



This **Properties & Connections** tab is divided as follows:

- **Instrument Properties** area — Provides useful hardware and software version information.
- **Matrix Connections** area — Provides switch-matrix connection information.
- **Self Test** button — Accesses the built-in VPU **Self Test** diagnostic utility.

These three areas are described below.

Instrument Properties area

The **Instrument Properties** area provides a variety of useful hardware and software version information, as well as calibration information.

Matrix Connections area

The **Matrix Connections** area displays the matrix connections that are associated with the VPU output terminals, if the following conditions apply:

- A matrix is included in the system configuration.
- The VPU is connected to the matrix.

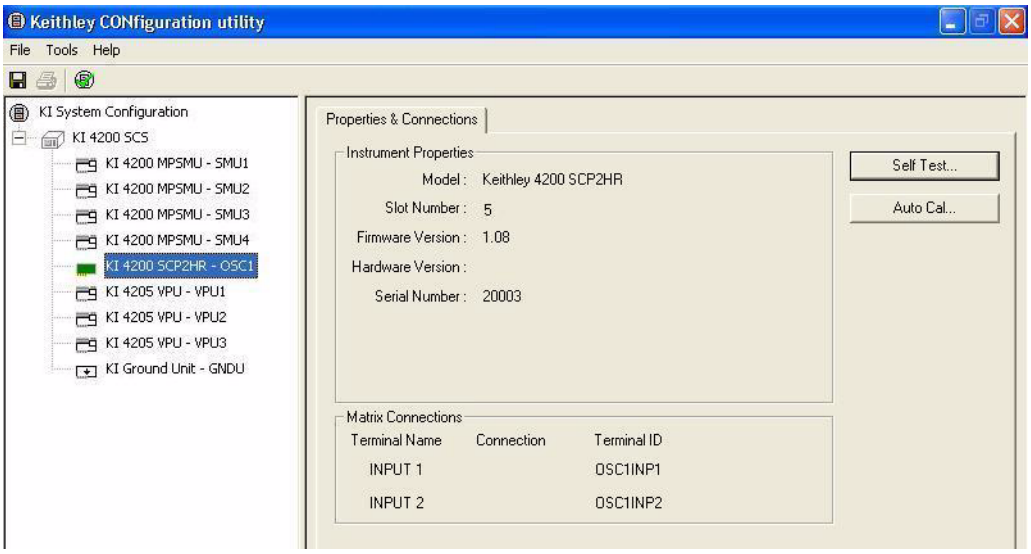
Self Test button

The **Self Test** button starts the **Self Test** utility, which aids in troubleshooting potential VPU hardware problems. When you run **Self Test,** the VPU performs several internal performance checks to determine if it is operating correctly. No external equipment is required. When the **Self Test** process finishes, a pass/fail window appears.

KI 4200 SCOPE Properties and Connections tab

When you select **KI 4200 SCOPE** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-19.

Figure 7-19
KI 4200 SCOPE Properties & Connections tab



This **Properties & Connections** tab is divided as follows:

- **Instrument Properties** area — Provides useful hardware and software version information.
- **Matrix Connections** area — Provides switch-matrix connection information.
- **Self Test** button — Accesses the built-in SCOPE **Self Test** diagnostic utility.
- **Auto Cal** button — Performs an auto calibration on the scope.

These four areas are described below:

Instrument Properties area

The **Instrument Properties** area provides a variety of useful hardware and software version information, as well as calibration information.

Matrix Connections area

The **Matrix Connections** area displays the matrix connections that are associated with the SCOPE input terminals, when the following conditions apply:

- A matrix is included in the system configuration.
- The SCOPE is connected to the matrix.

Self Test button

The **Self Test** button starts the **Self Test** utility, which aids in troubleshooting potential SCOPE hardware problems. When you run **Self Test,** the SCOPE performs several internal performance checks to determine if it is operating correctly. No external equipment is required. When the **Self Test** process finishes, a pass/fail window appears.

Auto Cal button

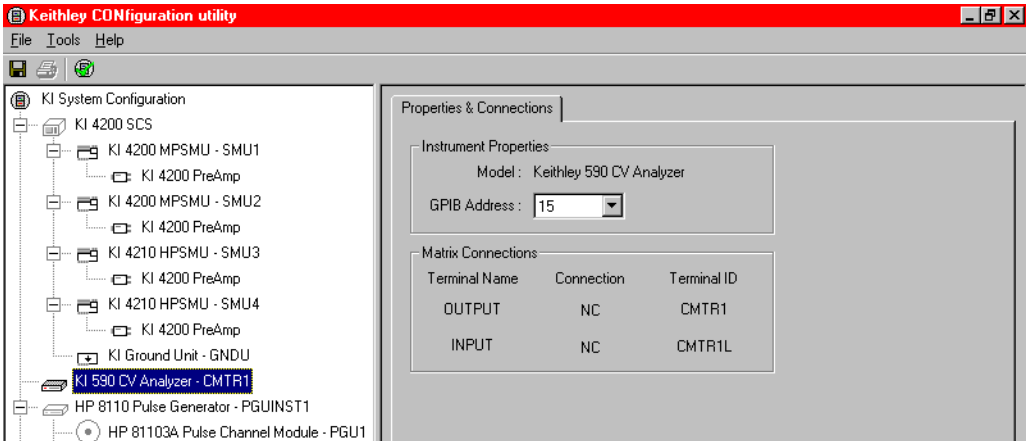
To ensure optimum measurement performance, use the **Auto Cal** button to calibrate the scope. No external equipment is required. The Cal routine compensates for internal temperature changes that may occur during extended use.

KI590 CV Analyzer Properties and Connections tab

When you select **KI 590 CV Analyzer** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-20.

NOTE *The Keithley Instruments Model 590 is a supported external instrument. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

Figure 7-20
KI 590 CV Analyzer Properties & Connections tab



This **Properties & Connections** tab provides access to the Keithley Instruments Model 590 instrument properties, and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following Keithley Instruments Model 590 instrument properties:

- **Model** Full vendor name, model number, and instrument description.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration via the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration independent manner. For more information, refer to "Keithley User Library Tool (KULT)" in Section 8.*

Matrix Connections area

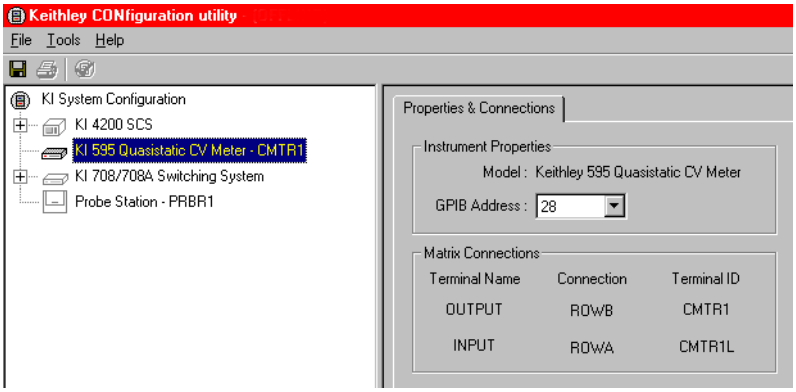
When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the Model 590 measurement terminals.

KI595 CV Analyzer Properties and Connections tab

When you select **KI 595 CV Analyzer** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-21.

NOTE *The Keithley Instruments Model 595 is a supported external instrument. The Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

Figure 7-21
KI 595 Quasistatic CV Meter Properties & Connections tab



This **Properties & Connections** tab provides access to the Keithley Instruments Model 595 instrument properties, and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following Keithley Instruments Model 595 instrument properties:

- **Model** Full vendor name, model number, and instrument description.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration by using the LPTLib getinstattr function. Proper usage of getinstattr allows you to develop user libraries in a configuration independent manner. For more information, refer to* Keithley User Library Tool (KULT) in Section 8.

Matrix Connections area

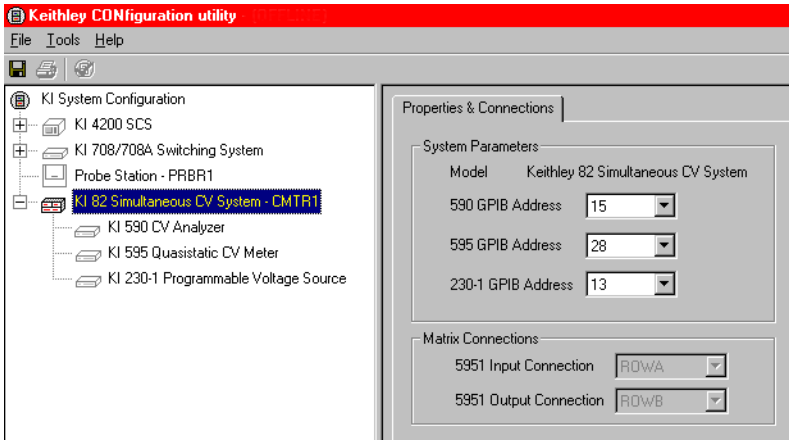
When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the Model 595 measurement terminals.

KI82 CV System Properties and Connections tab

When you select **KI 82 Simultaneous CV System** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-22.

NOTE *The Keithley Instruments Model 82 is a system of supported external instruments. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules (refer to Table 7-1 for additional supporting information). The Model 4200-SCS also provides sample projects for use with the Model 82 system.*

Figure 7-22
KI 82 Simultaneous CV System Properties & Connections tab



This **Properties & Connections** tab provides access to Keithley Instruments Model 82 system properties, and also provides useful switch-matrix connection information.

System Parameters area

The **System Parameters** area of this **Properties & Connections** tab provides access to the following Keithley Instruments Model 82 system properties:

- **Model** Full vendor name, model number, and instrument description.
- **GPIB Address** A primary GPIB address pull-down selection menu for each system instrument. Addresses that are in use display with adjacent asterisks (*). The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point (!) displays next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration by using the LPTLib getinstattr function. Proper use of getinstattr allows you to develop user libraries in a configuration-independent manner. For more information, refer to Keithley User Library Tool (KULT)* in Section 8.

Matrix Connections area

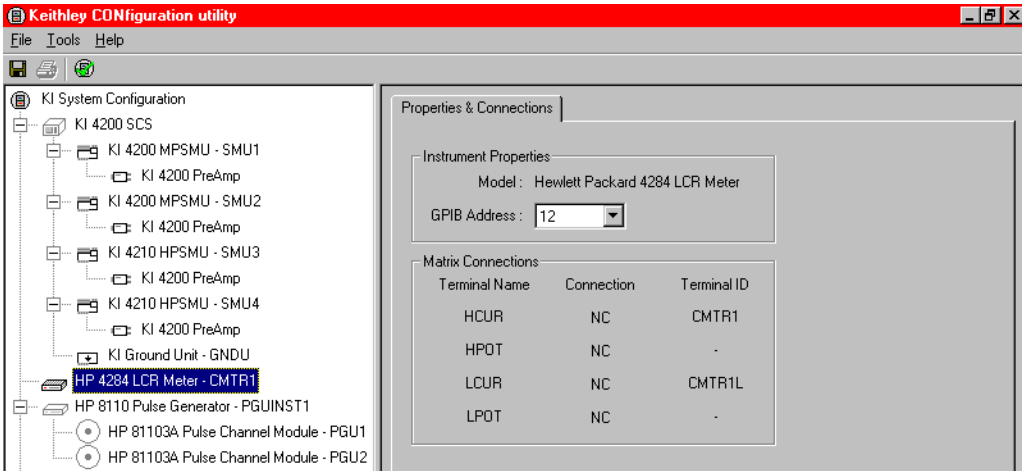
When the system configuration includes a matrix, the **Matrix Connections** area of the **Properties & Connections** tab displays the matrix connections that are associated with the measurement terminals of the system's Keithley Instruments Model 5951 Remote Input Coupler.

HP 4980 LCR Meter Properties and Connections tab

When you select **HP 4980 LCR Meter** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-23.

NOTE *The Agilent (HP) 4980 is a supported external instrument. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

Figure 7-23
HP 4980 LCR Meter Properties & Connections tab



This **Properties & Connections** tab provides access to the HP 4980 instrument properties, and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following HP 4980 instrument properties:

- **Model** Full vendor name, model number, and instrument description.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration via the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-*

independent manner. For more information, refer to Keithley User Library Tool (KULT) in Section 8.

Matrix Connections area

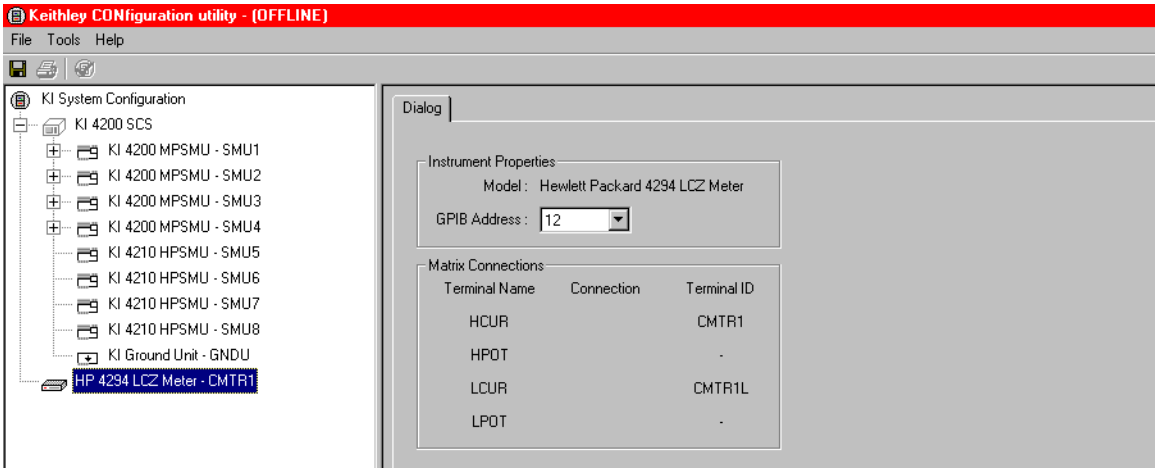
When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the Agilent 4980 measurement terminals.

HP 4294 LCR Meter Properties and Connections tab

When you select **HP 4294 LCR Meter** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-24.

NOTE *The* Agilent *(HP) 4294 is a supported external instrument. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

Figure 7-24
HP 4294 LCR Meter Properties & Connections tab



This **Properties & Connections** tab provides access to the HP 4294 instrument properties, and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following HP 4294 instrument properties:

- **Model** Full vendor name, model number, and instrument description.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration via the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-independent manner. For more information, refer to Keithley User Library Tool (KULT) in Section 8.*

Matrix Connections area

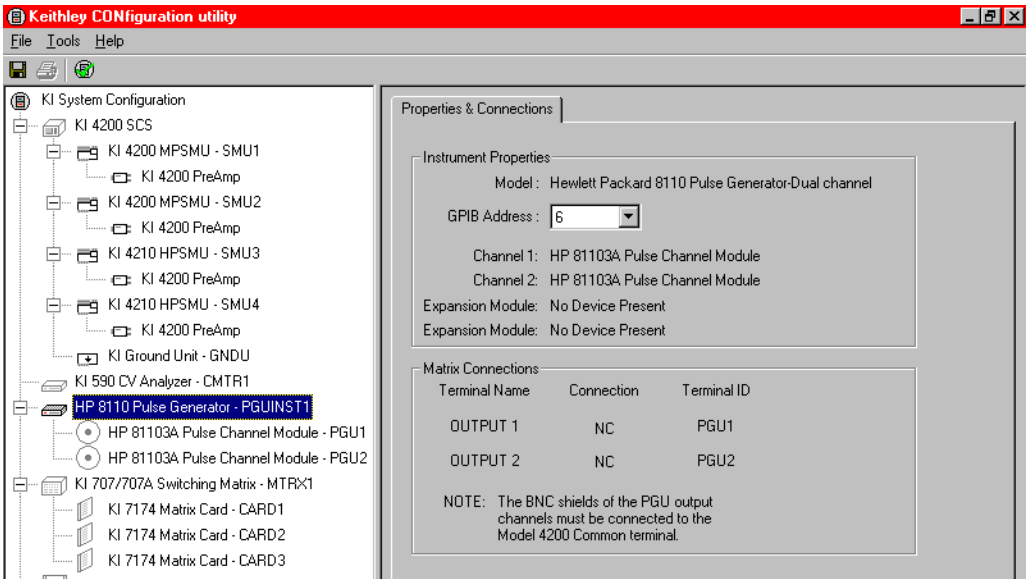
When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the Agilent 4294 measurement terminals.

HP 8110 and HP 81110 Properties and Connections tabs

When you select **HP 8110 Pulse Generator** in the Configuration Navigator, its **Properties & Connections** tab displays in the Workspace. See Figure 7-25.

NOTE *The Agilent (HP) 8110/81110 is a supported external instrument. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

Figure 7-25
HP 8110 Pulse Generator Properties & Connections tab



This **Properties & Connections** tab provides access to the HP 8110/81110 instrument properties, and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following HP 8110/81110 instrument properties:

- **Mode** Full vendor name, model number, and instrument description.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the

Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation-point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration via the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-independent manner. For more information, refer to* *Keithley User Library Tool (KULT)* in Section 8.

Matrix Connections area

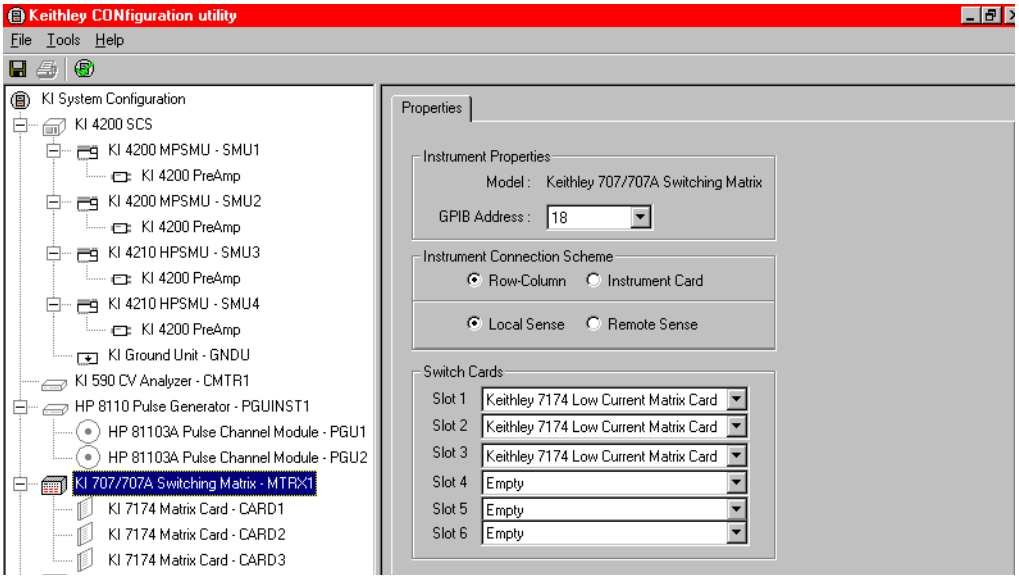
When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the Agilent 8110/81110 terminals.

KI 70X Switching Matrix Properties tab

NOTE *For general information about switch matrices, refer to "Using Switch Matrices" in Appendix B.*

Selecting the **KI 707/707A Switching Matrix** in the Configuration Navigator causes its **Properties** tab to be displayed in the Workspace. See Figure 7-26.

Figure 7-26
KI 707/707A Switching Matrix Properties tab



A nearly identical (and functionally equivalent) **Properties** tab is displayed in the Workspace when a **KI 708/708A Switch Matrix** is selected. The only difference between the Keithley Instruments Model 707S and 708 Switch Matrices is that the Models 707 has six matrix-card slots, and the Model 708 has one. The Model 707 is used for illustration purposes.

NOTE *The Keithley Instruments Models 707 and 708 are supported external instruments. Therefore, the Model 4200-SCS provides a user library containing preconfigured data acquisition and control user modules. Refer to Table 7-1 for additional supporting information.*

A KI 707/707A or KI 708/708A **Properties** tab contains the following:

- **Instrument Properties** area — Displays the Keithley Instruments Model 707/708 instrument properties.
- **Instrument Connection Scheme** area — Allows you to specify the instrument connection scheme:
  - Whether a row-column or instrument-card connection scheme is used
  - Whether the instrument SENSE terminals are used (whether remote voltage sensing is used for measurements)

NOTE *Specify physical instrument-to-card and card-to-prober/fixture connections via a different tab (the appropriate KI 7XXX matrix-card **Properties** tab). Refer to KI 7XXX Matrix Card Properties tab in Section 7.*

- **Switch Cards** area — Allows you to specify, by model number, the KI 7XXX-series matrix card that is installed in each slot of the Keithley Instruments Model 707/708.

Each **Properties** tab area is discussed separately below.

NOTE *These and other KCON switch matrix settings result in simplified matrix connections. You need to do the following only initially:*

- *Configure **Instrument Connection Scheme** and **Switch Cards** areas as described below.*
- *Specify the physical instrument-to-card and card-to-prober/fixture connections, as described in KI 7XXX Matrix Card Properties tab in Section 7.*
- *Physically make the specified instrument-to-card and card-to-prober/fixture connections.*

Thereafter, you can specify instrument-to-prober/fixture connections simply by specifying the corresponding terminal and prober/fixture pins in a KITE UTM. You need not specify matrix cross points. The Model 4200-SCS automatically routes the signals through the matrix.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following instrument properties:

- **Model** — Full vendor name, model number, and instrument description.
- **GPIB Address** — Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the

configuration, a red exclamation point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration via the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-independent manner. For more information, refer to* *Keithley User Library Tool (KULT)* in Section 8.

Instrument Connection Scheme area

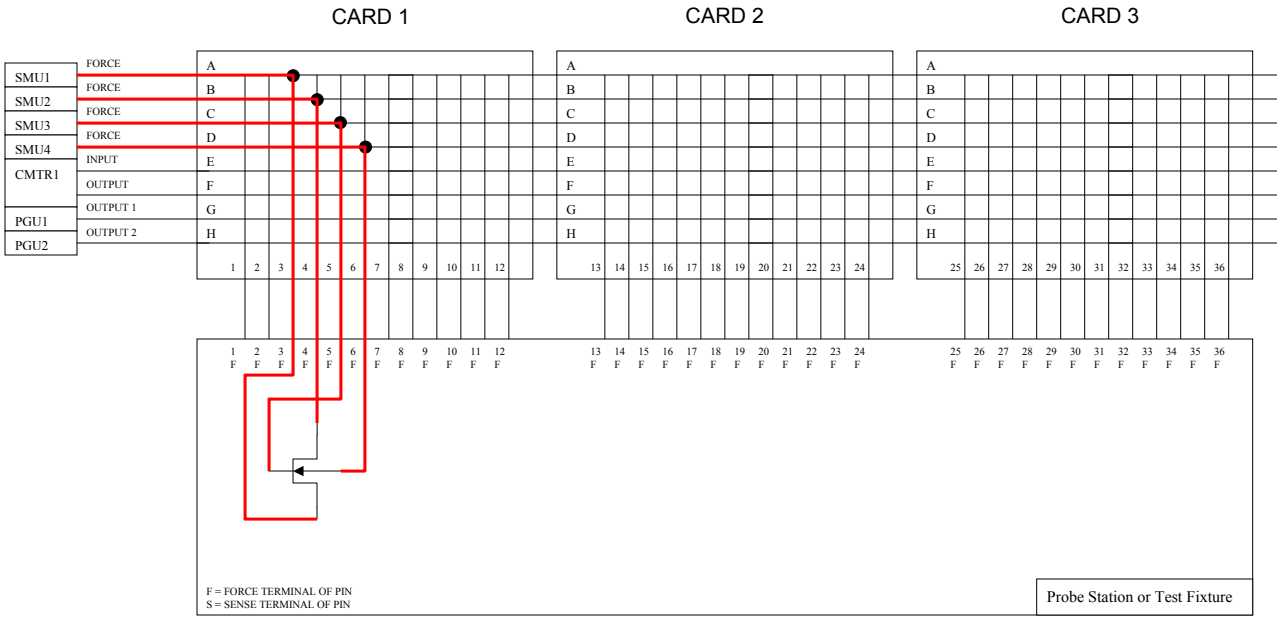
The following **Instrument Connection Scheme** selections define the *scheme* for interconnections between the instruments, the switch-matrix rows and columns, and the test system (prober or test fixture).

- **Row-Column** (instruments to rows and prober/test fixture to columns) or **Instrument Card** (both instrument and prober/test fixture to columns)
- **Local Sense** (connections only to instrument FORCE terminals) or **Remote Sense** (connections both to instrument FORCE and SENSE terminals)

These selections are discussed below in more detail:

- **Row-Column** With the **Row-Column** scheme, instruments are connected to switch-matrix rows, and prober/test fixture pins are connected to switch-matrix columns. This connection scheme is the simplest. Instrument signals can route to prober/test-fixture pins through only one matrix card, as illustrated in Figure 7-27. However, the **Row-Column** scheme limits the number of external instruments. If you need to connect numerous external instruments to the prober/test-fixture, use the **Instrument Card** scheme.

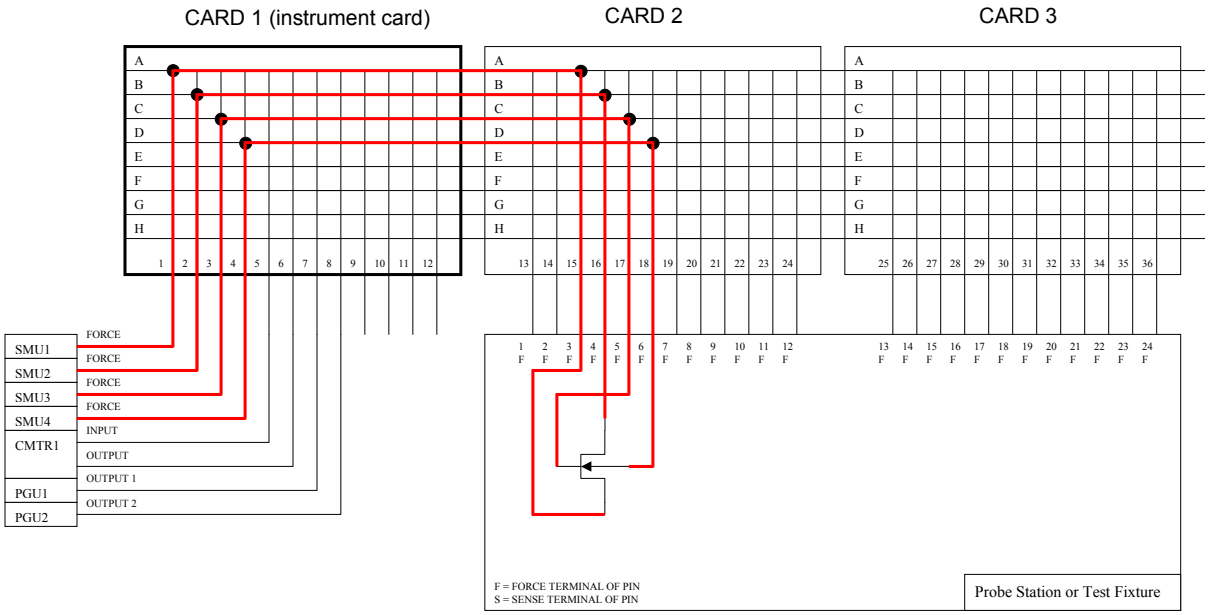
Figure 7-27
Row-Column, Local Sense Connection Scheme example



- **Instrument Card** With the **Instrument Card** scheme, both instruments and prober/test-fixture pins are connected to the columns of the switch matrix.

Instrument signals route to the prober/test-fixture pins through two or more matrix cards, as illustrated in Figure 7-28. This connection scheme can support large systems with numerous instruments by removing the eight-row instrument connection limitation.

Figure 7-28
Instrument Card, Local Sense Connection Scheme example



- **Local Sense** Use **Local Sense** when the measurement-pathway resistance is small and the associated voltage errors are negligible. The measurement pathway is comprised of the following conductors, connected in series:

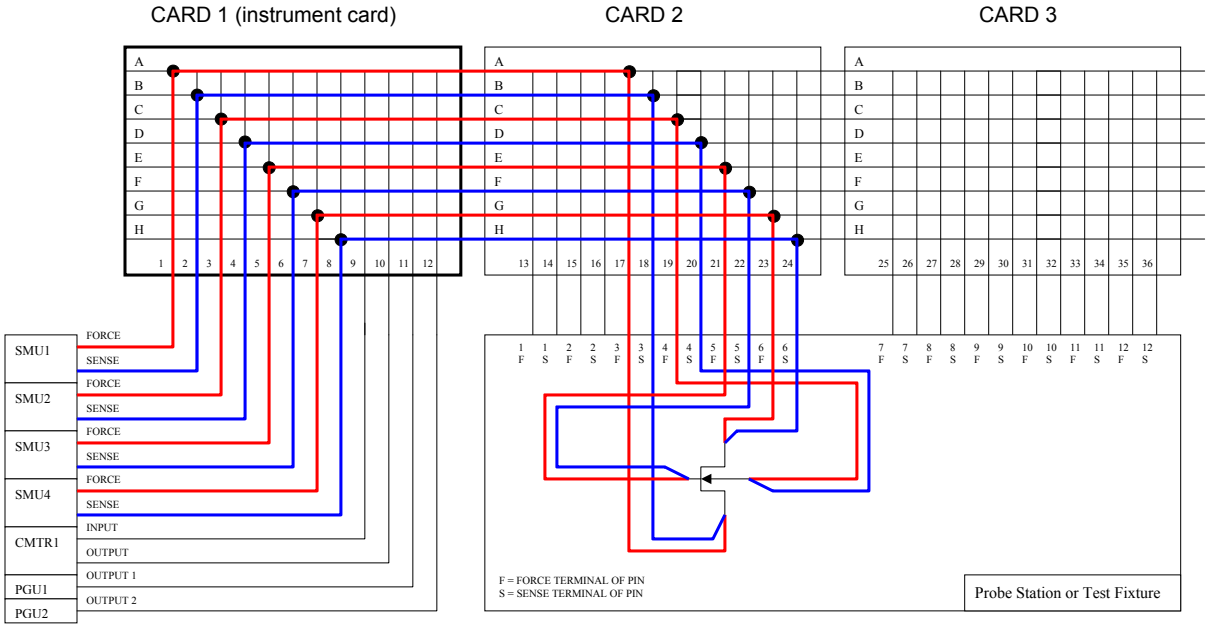
  - The cables used to connect the instruments to the matrix
  - The internal matrix-card signal path
  - The cables used to connect the matrix to the prober or test fixture

  Current flowing through the measurement pathway creates a voltage drop (an error voltage) that is directly proportional to the pathway resistance. This error voltage is present in all **Local Sense** voltage measurements.

NOTE *For more information regarding **Local Sense** and **Remote Sense**, refer to "Remote sensing" in Section 5.*

- **Remote Sense** Use **Remote Sense** to eliminate the effects of measurement pathway resistance. Figure 7-29 illustrates the use of **Remote Sense** in an instrument card configuration. Note that **Remote Sense** requires twice as many measurement pathways. The FORCE pathways (in red) are the current-carrying pathways, and the SENSE pathways (in blue) are the measurement pathways.

Figure 7-29
Instrument Card, Remote Sense Connection Scheme example



Switch Cards area

The **Switch Cards** area of a KI 707/707A or KI 708/708A **Properties** tab is used to specify which matrix card is installed in each slot. Use the pull-down menu to select each card.

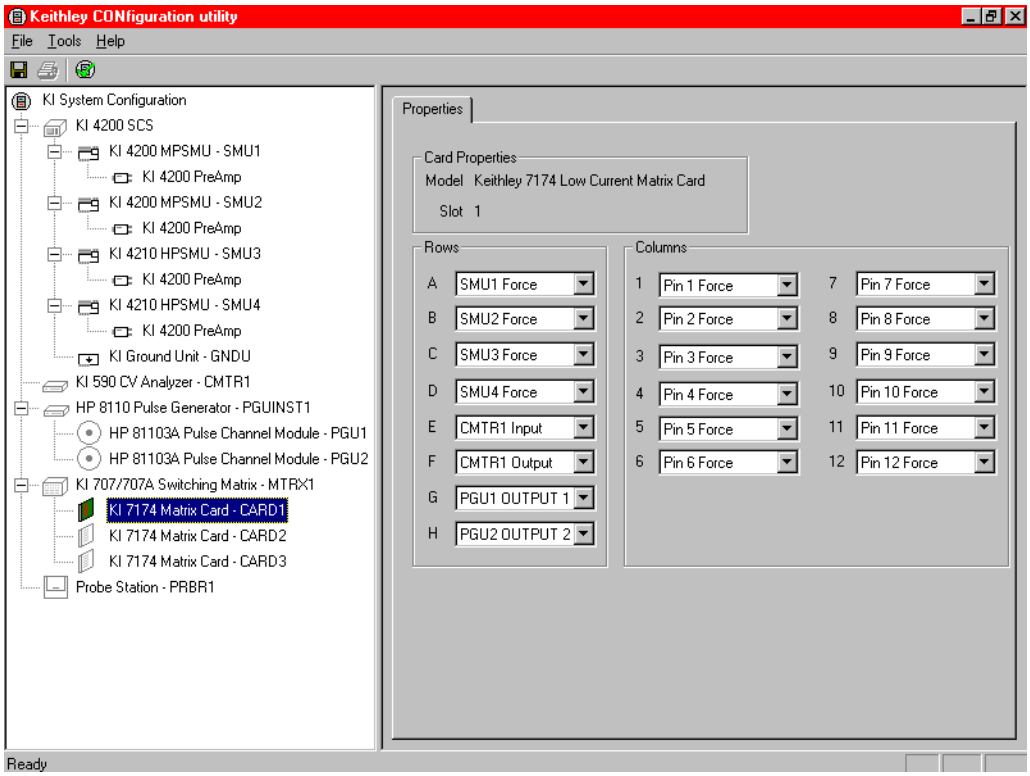
NOTE *The Model 4200-SCS does not support mixed-card configurations. All cards must be the same type.*

KI 7XXX Matrix Card Properties tab

NOTE *For general information about switch matrices, refer to Using Switch Matrices in Appendix B.*

When you select a **KI 7XXX Matrix Card** in the Configuration Navigator, its **Properties** tab displays in the Workspace. Figure 7-30 shows the **Properties** tab that displays when you select a **KI 7174 Matrix Card**.

Figure 7-30
KI 7174 Matrix Card Properties tab



Nearly identical (and functionally equivalent) **Properties** tabs display in the Workspace for other KI 7XXX matrix cards that are supported by the Keithley Instruments Model 707/708.

NOTE *Detailed signal routing and physical matrix connection information for each supported matrix card is provided in "Controlling a switch matrix" in Section 2 of the Applications Manual.*

The two areas of a **Properties** tab define the following connections for a switch-matrix card:

- Between the measurement instrumentation and the matrix card
- Between the matrix card and the test system (prober or test fixture)

Figure 7-30 shows the KI 7XXX matrix-card **Properties** tab configuration that is required to support the physical connection configuration that is shown in Figure 7-27.

Card Properties area

The **Card Properties** area describes the following:

- **Model** Full vendor name, model number, and card description
- **Slot** Switch matrix slot that the matrix card is installed in

Rows area

In the **Rows** area, the combo boxes labeled **A** through **H** correspond to the eight rows of all Keithley Instruments Model 707/708 compatible matrix cards. Use the pull-down menus of the combo boxes to connect the rows to various instrument terminals.

NOTE *Prober or test-fixture pins are always connected to matrix-card columns.*

Columns area

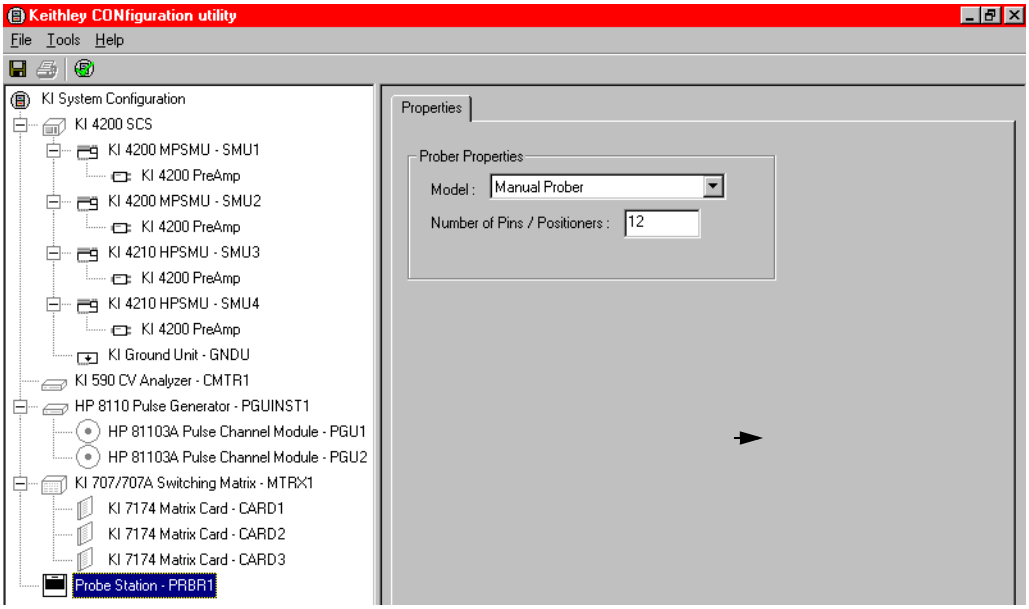
In the **Columns** area, the combo boxes labeled **1** through **12** correspond to the 12 columns of all Keithley Instruments Model 707/708 compatible matrix cards. Use the pull-down menus of the combo boxes to connect the columns to various instrument terminals and/or prober/test-fixture pins.

NOTE *You may connect instrument terminals to the matrix card columns only when the **Instrument Connection Scheme** setting is active (refer to Instrument Connection Scheme area earlier in this section).*

Probe Station Properties tab

Selecting the **Probe Station** in the Configuration Navigator causes its **Properties** tab to be displayed in the Workspace. See Figure 7-31.

Figure 7-31
Prober Properties tab



As for other external instruments, the Model 4200-SCS controls a probe station by using KULT user modules (when connected to User Test Modules (UTMs) in a KITE project). The Model 4200-SCS comes with the `prbgen` library of prober control user modules.The `prbgen` user library, developed and maintained by Keithley Instruments, is generic, thereby allowing KITE to control all supported probers in the same manner. KITE projects utilizing `prbgen` work with any Keithley Instruments supported prober.

When using a switch matrix, one probe station or one test fixture must be present in the system configuration. This is necessary because the probe station or test fixture establishes the number of test system pins. The matrix is cabled to the test system pins, and instrument terminals are routed through the matrix to the pins using the `matrixulib` user modules. Refer to Using Switch Matrices in Appendix B for more information.

NOTE *Refer to Table 7-1 for additional supporting information.*

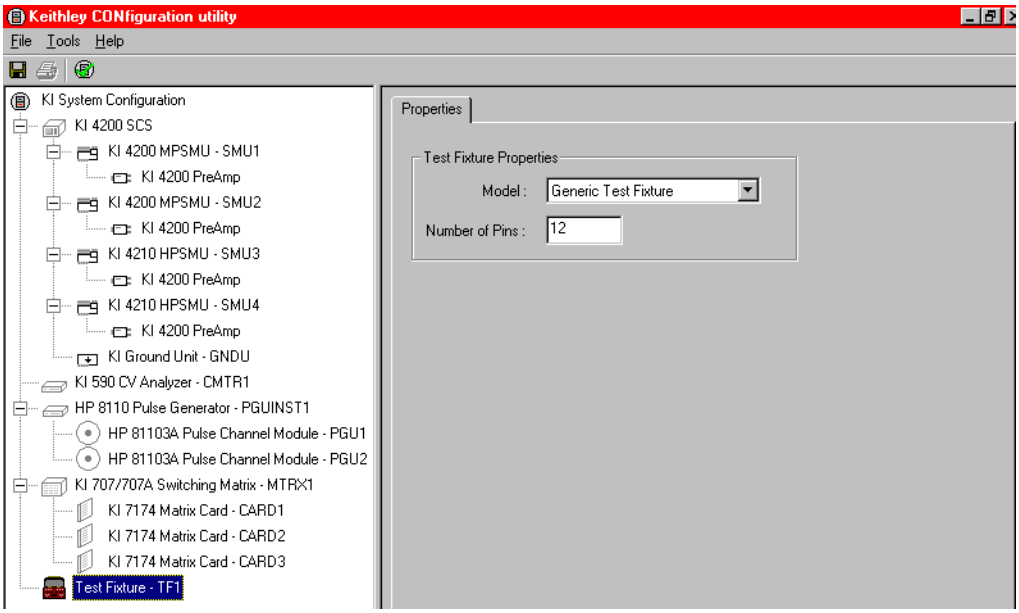
Two settings are provided, as follows:

- **Model** A pull-down menu listing the supported probers
- **Number of Pins/Positioners** Edit box for defining the number of probe-card or positioner pins: the minimum number of pins is 2; the maximum is 72

Test Fixture Properties

Selecting the **Test Fixture** in the Configuration Navigator causes its **Properties** tab to be displayed in the Workspace. See Figure 7-32.

Figure 7-32
Test Fixture Properties



When using a switch matrix, one probe station or one test fixture must be present in the system configuration, because the probe station or test fixture establishes the number of test-system pins. The matrix is cabled to the test system pins, and instrument terminals are routed through the matrix to the pins using the `matrixulib` user modules. Refer to Using Switch Matrices in Appendix B for more information.

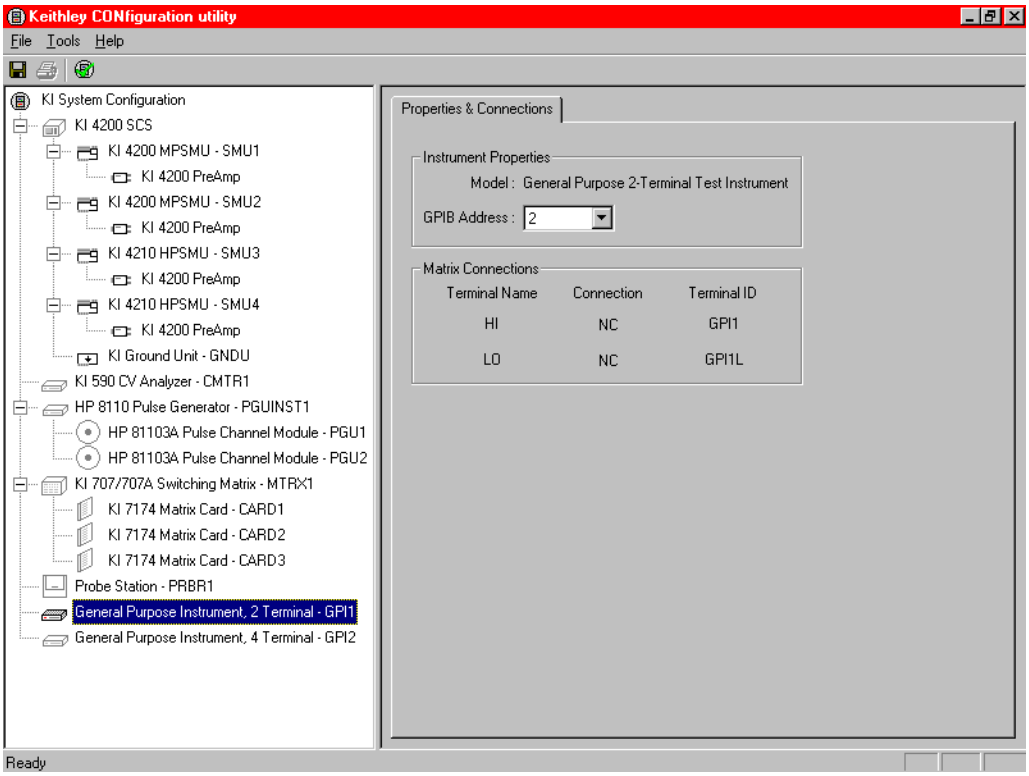
Two settings are provided, as follows:

- **Model** A pull-down menu listing the supported test fixtures
- **Number of Pins** Edit box for defining the number of test fixture pins: the minimum number of pins is 2; the maximum is 72

General Purpose Instrument, 2-Terminal Properties and Connections tab

Selecting **General Purpose Instrument, 2-Terminal** in the Configuration Navigator causes its **Properties & Connections** tab to be displayed in the Workspace. See Figure 7-33.

Figure 7-33
General Purpose Instrument, 2-Terminal, Properties & Connections tab



Add a General Purpose Instrument (GPI) to your system configuration when your application requires an *unsupported* external instrument.

NOTE *An unsupported external instrument is an instrument that cannot be added to the system configuration by selecting it from the supported* ***Tools > Add External Instrument*** *categories.*

A two-terminal GPI is an unsupported external instrument with 2-terminals (HI and LO) (for example, a current source). Generally, the GPI HI and LO terminals transmit or receive the GPI stimulus signals.

To control the operation of an IEEE-488 or RS-232 GPI in a KITE project, create a user library with KULT and use the LPTLib I/O functions (`kib*` and `ksp*`) to communicate with the GPI (refer to Keithley User Library Tool (KULT) in Section 8, for more information). However, as for any other instrument in the system configuration, GPI terminals can be automatically routed to test system pins using the `ConnectPins` user module in the provided `matrixulib` user library (refer to Using Switch Matrices in Appendix B for more information).

This **Properties & Connections** tab provides access to GPI instrument properties and also provides useful switch-matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following two-terminal GPI properties:

- **Model** Indicates that the instrument is a two-terminal GPI.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration by using the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-independent manner. For more information, refer to Keithley User Library Tool (KULT) in Section 8.*

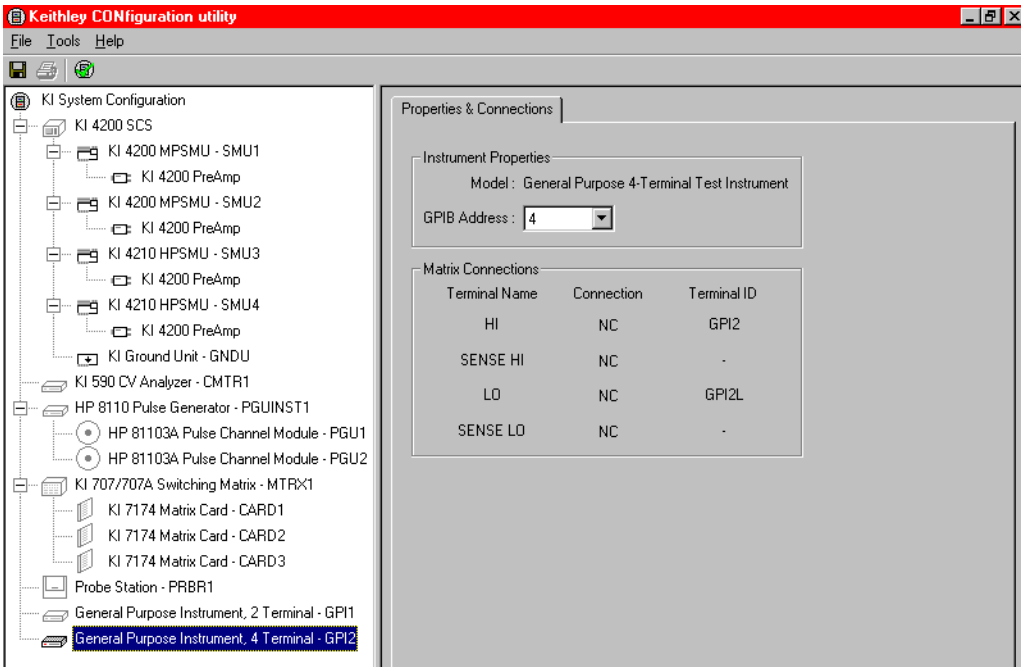
Matrix Connections area

When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the 2-terminal GPI terminals.

General Purpose Instrument, 4-Terminal Properties and Connections tab

Selecting **General Purpose Instrument, 4-Terminal** in the Configuration Navigator causes its **Properties & Connections** tab to be displayed in the Workspace. See Figure 7-34.

Figure 7-34
General Purpose Instrument, 4-Terminal, Properties & Connections tab



Add a General Purpose Instrument (GPI) to your system configuration when your application requires an *unsupported* external instrument.

NOTE *An unsupported external instrument is an instrument that cannot be added to the system configuration by selecting it from the supported* **Tools > Add External Instrument** *categories.*

A four-terminal GPI is an unsupported external instrument with four terminals (HI, SENSE HI, LO, and SENSE LO), for example, a digital multimeter. Generally, the GPI HI and LO terminals transmit or receive the GPI stimulus signals. The GPI SENSE HI and SENSE LO terminals typically measure the DUT (device under test) response to the GPI stimulus.

NOTE *The SENSE HI and SENSE LO signals are automatically routed through separate (parallel) switch matrix pathways when making connections between the GPI HI/LO terminals and the test-system pins.*

To control the operation of an IEEE-488 or RS-232 GPI in a KITE project, create a user library with KULT and use the LPTLib I/O functions (`kib`* and `ksp`*) to communicate with the GPI (refer to Keithley User Library Tool (KULT) in Section 8, for more information). However, as for any other instrument in the system configuration, GPI terminals can be automatically routed to test system pins using the `ConnectPins` user module in the provided `matrixulib` user library (refer to Using Switch Matrices in Appendix B, for more information).

This **Properties & Connections** tab provides access to the GPI instrument properties and also provides useful switch matrix connection information.

Instrument Properties area

The **Instrument Properties** area of this **Properties & Connections** tab provides access to the following 4-terminal GPI properties:

- **Model** Indicates that the instrument is a four-terminal GPI.
- **GPIB Address** Primary GPIB address pull-down selection menu. Addresses that are in use are displayed with asterisks (*) next to them. The minimum address value is 0; the maximum is 30 (GPIB address 31 is reserved as the Model 4200-SCS controller address). If the selected GPIB address conflicts with the GPIB address of another instrument in the configuration, a red exclamation point symbol (!) is displayed next to the address.

NOTE *You can programmatically read the GPIB address and other instrument properties from the system configuration by using the LPTLib* `getinstattr` *function. Proper usage of* `getinstattr` *allows you to develop user libraries in a configuration-independent manner. For more information, refer to Keithley User Library Tool (KULT)* in Section 8.

Matrix Connections area

When a matrix is included in the system configuration, the **Matrix Connections** area of this **Properties & Connections** tab displays the matrix connections that are associated with the 4-terminal GPI terminals.

Section 8

Keithley User Library Tool (KULT)

In this section:

Keithley User Library Tool (KULT)

NOTE *The Microsoft Visual Studio C++ compiler is used to create, modify, and debug KULT modules (user library and user modules). When ordered with a Keithley Model 4200-SCS system, the software is installed at the factory.*

Introduction

The Keithley User Library Tool (KULT) is a tool used to create and manage *user libraries*. A user library is a collection of one or more *user modules*. User modules are C programming language subroutines (or functions). User libraries are created to control instrumentation, analyze data, or perform any other system automation task programmatically. Once a user library has been successfully built using KULT, its user modules can be executed using the Keithley Interactive Test Environment *(*KITE*)* software tool.

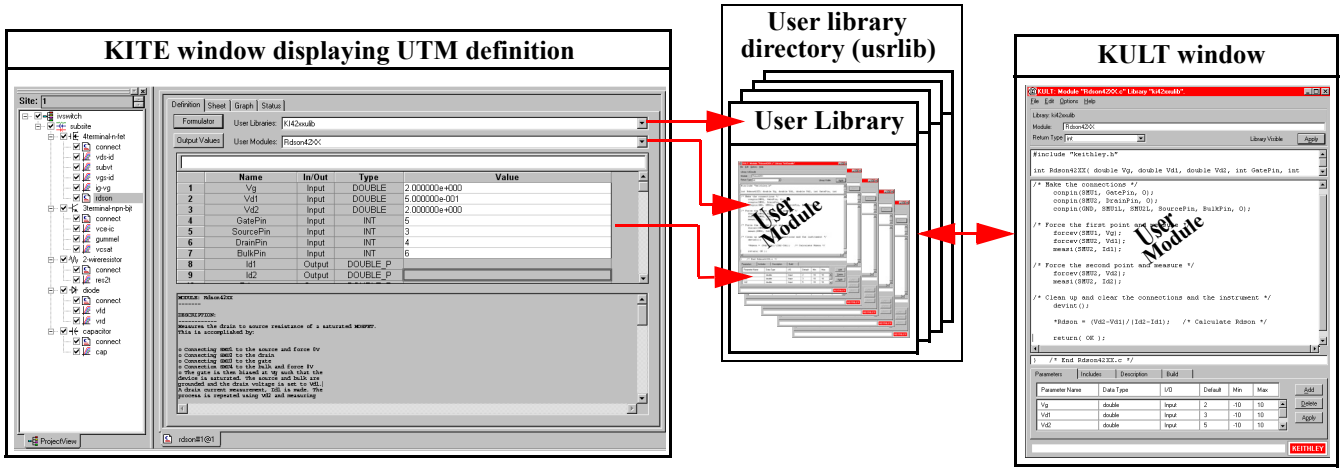
KULT provides a simple graphical user interface that helps even a novice programmer to effectively enter code, compile a user module, and link (build) a user library. KULT also provides user-library management features, including the copy module, copy library, delete module, and delete library menu commands. KULT manages user libraries in a structured manner. You can create your own user libraries to extend the capabilities of the Model 4200-SCS without requiring a software upgrade from Keithley.

To execute a KULT user module in KITE, you create a KITE user test module (UTM) and connect it to the user module. Once this user module is connected to the UTM, the following occurs each time KITE executes the UTM:

- KITE dynamically loads the user module and the appropriate user library directory (usrlib).
- KITE passes the user-module parameters (stored in the UTM) to the user module.
- Data generated by the user module is returned to the UTM for interactive analysis.

Figure 8-1 illustrates the relationships between user libraries, user modules, UTMs, KITE, and KULT.

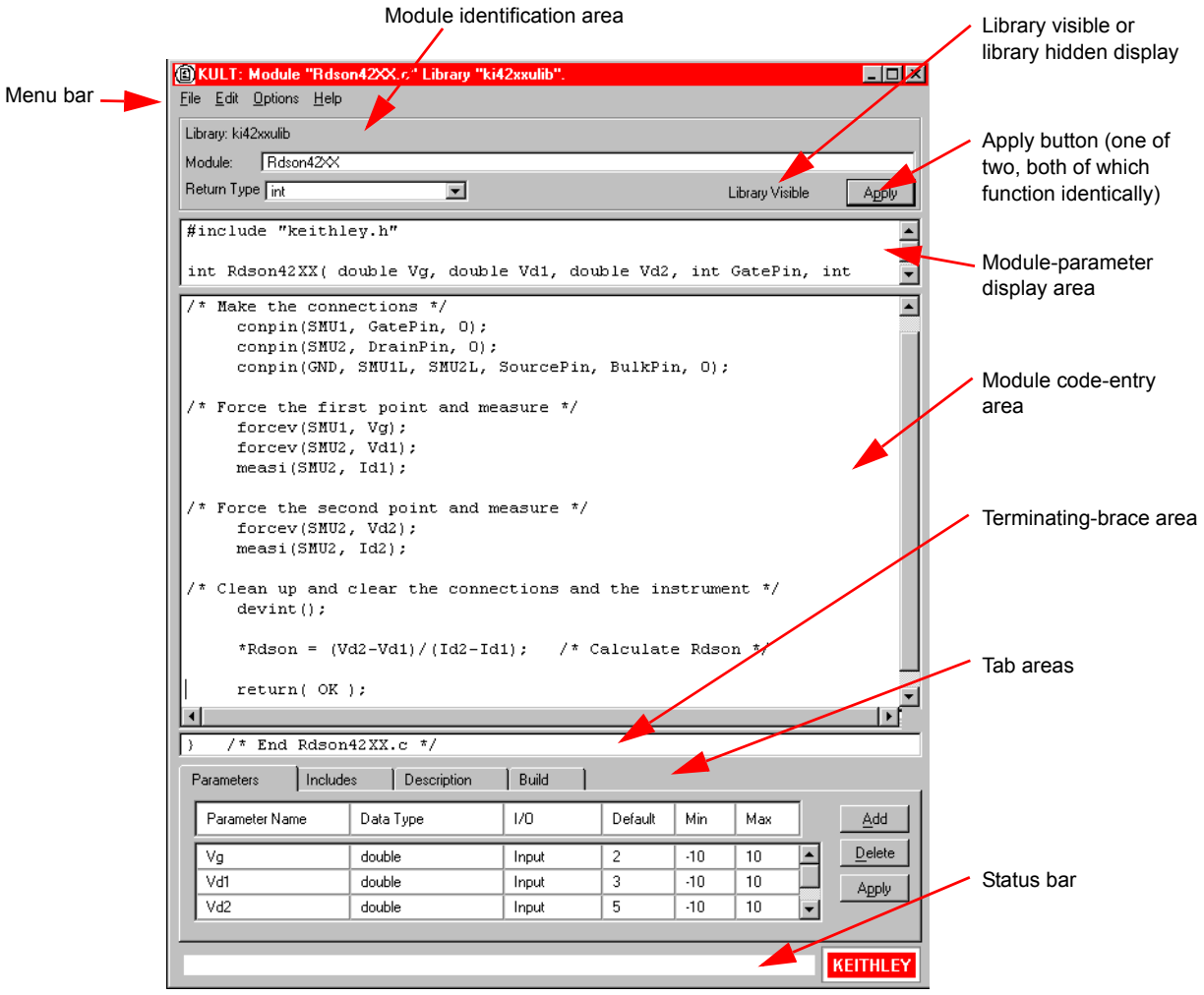
Figure 8-1
Relationships between KULT, KITE, user libraries, and UTMs



KULT window

The KULT window is shown in Figure 8-2. It provides all the menus, controls, and user-entry areas needed to create/edit/view and build a user library, and to create/edit/view and compile a user module.

Figure 8-2
KULT window



Each feature of the KULT window is discussed in the following subsections.

Understanding the module identification area

The module identification area is located directly below the menu bar and defines the presently open user library and user module. The components of this area are used as follows:

- **Library Name**: Displays the name of the presently open (active) user library. To specify a user library, use the **File > New Library** or the **File > Open Library** command (see "File menu" on page 8-12 for more information).
- **Module Name**: Displays the name of the presently open user module. To specify a user module, use the **File > New Module** or the **File > Open Module** command (see "File menu" on page 8-12 for more information). Module Name also

allows you to create a copy of the presently open user module in the same user library:

1. Enter a new name in the **Module:** text box
2. Click **Apply**.

NOTE *When naming a user module, remember to conform to case-sensitive C programming language naming conventions. Do not duplicate names of existing user modules or user libraries.*

- **Return type**: Defines the data type of all codes that are returned by `return(code)` statements in the user module. A scroll box, in which you select one of the following variable types:
  - **char**: Character data
  - **double**: Double precision data
  - **float**: Single precision, floating point data
  - **int**: Integer data
  - **long**: 32-bit integer data
  - **void**: No data returned

NOTE *When a UTM is executed by KITE, the value of the* `return(code)` *statement is displayed on the Data worksheet in the column labeled with the module name.*

Library Visible / Library Hidden: Displays whether or not the presently open user library is available to KITE. To change the hidden/visible status, check or uncheck the **Hide Library** option in the **Options** menu (Figure 8-11).

Apply: Used to update the presently open user module to reflect additions and changes. Also, if you change the **Module Name** for the presently open user module, clicking the **Apply** button creates a copy of the module under the new name.

Understanding the module parameter display area

The module parameter area is a display-only area that is located directly below the module identification area. In the module-parameter area, KULT displays the following:

- The C-language function prototype for the user module, reflecting the parameters that are specified in the Parameters tab area, and the `return(code)` data type.
- The `#include` and `#define` statements that are specified in the Includes tab areas.

Understanding the module code entry area

The module code-entry area is located below the module-parameter area. The module code-entry area is the KULT window location where you enter, edit, or view the user-module C-code. Scroll bars located to the right and below the module-code entry area let you move through the code.

NOTE ***Do Not*** *enter the following C-code items in the module code-entry area* (KULT *enters these at special locations based on information in other places in the* KULT *window):*

- `#include` *and* `#define` *statements*
- *The function prototype*
- *The terminating brace*

To control internal or external instrumentation, use functions from the Linear Parametric Test Library (LPTLib). For more information refer to the Reference Manual, LPT Library Function Reference, page 8-90.

Understanding the terminating brace area

The terminating-brace area is a display-only area. KULT automatically enters and displays the terminating-brace for the user module code.

Understanding the tab areas

Four tab areas are accessed by clicking one of the following tabs:

- Parameters
- Includes
- Description
- Build

Selections within the Parameters tab, Includes tab, and Description tab areas are facilitated using pop-up menus. The next four subsections describe the tab areas.

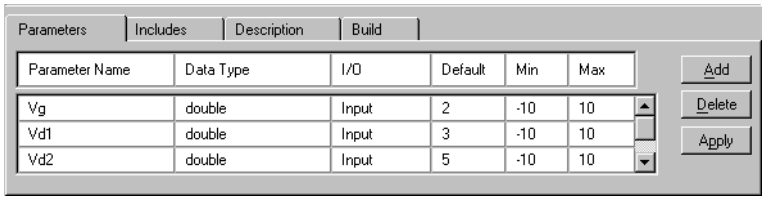
Parameters tab area

The Parameters tab area is used to define and display the following for each parameter that is included in the user module call:

- Parameter name
- Parameter data type
- Input or output (I/O) data direction
- Default, min, and max values for the parameter

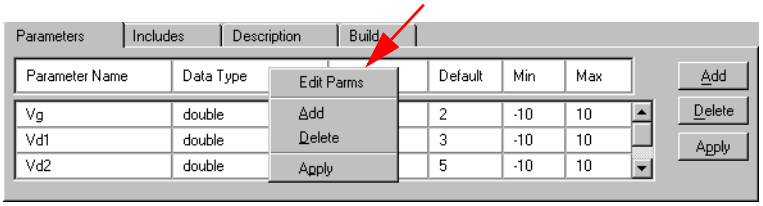
The Parameters tab area is located near the bottom of the KULT main screen. See the example in Figure 8-3.

Figure 8-3
Parameters tab area, example entries for the RDS on 42XX user module



A pop-up menu duplicates the Add, Delete, and Apply buttons at the right side of the Parameters tab area. Open the pop-up menu by right-clicking anywhere in the Parameters tab area (see Figure 8-4).

Figure 8-4
Parameters tab area Add, Delete, Apply pop-up menu



To add, delete, or apply a parameter:

- To add a parameter, click **Add** and then enter the information as indicated in the field descriptions that follow.
- To delete a parameter, first click the parameter name or any of the adjacent fields; then click **Delete**.
- To apply changes made in the Parameters tab area, click **Apply**.

The next four subsections describe the six fields of the Parameters tab area.

Parameter name field

The parameter name field identifies the parameters that are passed to the user module. That is, these are the same parameters that would be specified in the user-module function prototype (which KULT constructs from the Parameters tab entries when you click **Apply,** and then displays in the module-parameter display area).

Data type field

The data type field specifies the parameter data type. Clicking on the arrow at the right of the data type field activates a pop-up menu, which lists the following data types:

- **char** Character data
- **char*** Pointer to character data
- **float** Single precision, floating point data
- **float*** Pointer to single precision, floating point data
- **double** Double precision data
- **double*** Pointer to double precision point data
- **int** Integer data
- **int*** Pointer to integer data
- **long** 32-bit integer data
- **long*** Pointer to 32-bit integer data
- **F_ARRAY_T** Floating point array type
- **I_ARRAY_T** Integer array type
- **D_ARRAY_T** Double precision array type

I/O field

The I/O field defines whether the parameter is an input or output type. Clicking on the arrow to the right of the I/O field activates a pop-up menu that shows the Input and output selections.

NOTE *The pop-up menu is displayed, and output can be selected, only for pointer data types (char*, float*, double*, and so on.) and array data types (*`I_ARRAY_T, F_ARRAY_T, and D_ARRAY_T`*).*

Default, min, and max fields

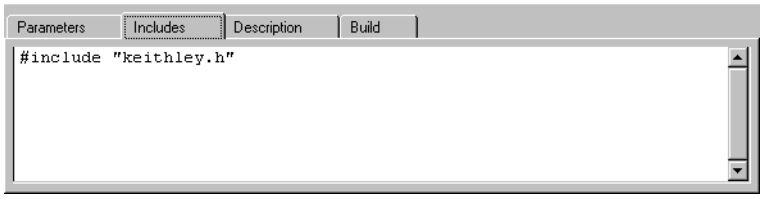
The Default, Min, and Max fields are used to specify the following:

- **Default field**: Specifies the default value for a non-array (only) input parameter.
- **Min field**: Specifies the minimum recommended value for a non-array (only) input parameter. When the user module is used in a KITE UTM, configuration of the UTM with a parameter value smaller than the min value causes KITE to display an out-of-range message (for a brief explanation of UTMs, refer to the Reference Manual, User test module (UTM), page 6-47).
- **Default field**: Specifies the maximum recommended value for a non-array (only) input parameter. When the user module is used in a KITE UTM, configuration of the UTM with a parameter value larger than the max value causes KITE to display an out-of-range message.

Includes tab area

The includes tab area lists the header files used within the user module. This area can be used to add `#include` and `#define` statements to the presently open user module. (see Figure 8-5)

Figure 8-5
Default Includes tab area



By default, KULT automatically enters the `keithley.h` header file into the includes tab area. The `keithley.h` header file includes the following frequently used C-programming interfaces:

- `#include<stdio.h>`
- `#include<stdlib.h>`
- `#include<string.h>`
- `#include<math.h>`
- `#include"windows.h"`

In most cases, it is not necessary to add items to the includes tab area, because `keithley.h` provides access to the most common C functions. However, in some cases, both of the following may apply:

- You do not wish to include `keithley.h`
- You wish to include only the header files specifically needed by your user module, and all of the user modules on which it depends.

If so, you must minimally include the following header files and `#define` statements to properly compile and build user modules and user libraries:

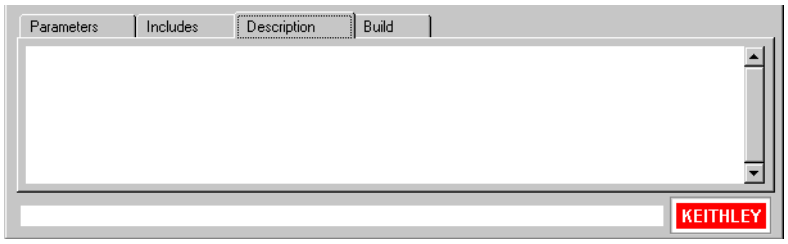
```
#include "lptdef.h"
#include "lptdef_lowercase.h"
#include "kilogmsg_proto.h
#include "ktemalloc.h"
#include "usrlib_proto.h"
```

```
#define PTexit _exit
#define exit Unsupported Syntax
#define abort Unsupported Syntax
#define terminate Unsupported Syntax
```

Description tab area

The description tab area, shown in Figure 8-6, allows you to enter descriptive information for the presently open user module. Information entered in this area documents the module to the KITE user and is used to create KITE user library help.

Figure 8-6
Description tab area



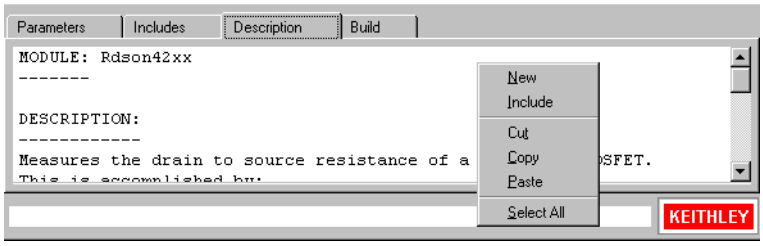
CAUTION **Do not use C-code comment designators (/*, */, or //) in the Description tab area. When the user-module code is compiled, KULT also evaluates the text in this area. C-code comment designators in the Description tab area can be misinterpreted, causing errors.**

NOTE *Do not place a period in the first column (the left-most position) of any line in the description tab area. Any text after a first-column period will not be displayed in the documentation area of a KITE UTM definition document.*

To enter a description:

1. Left-click in the **Description** tab area and enter the description.
2. Right-click in the **Description** tab area to open a pop-up edit menu. (see Figure 8-7)

Figure 8-7
Pop-up edit menu for the Description tab area



The pop-up edit menu commands in the description tab area are used as follows:

- **New**: Deletes the present description from the description tab area, allowing you to enter a new description.

- **Include**: Imports any file that you specify, typically a text file, into the document tab area: only (to import a `*.C` file into the module code-entry area. See "File menu" on page 8-12 information on the include command). Clicking **Include** displays the Include dialog box, in which you either browse and select a file or directly enter a file name and path. Clicking **Open** inserts the contents of the file at the cursor location.
- **Cut**: Removes highlighted text from the description tab area and copies it to the Windows® Clipboard®. The text on the clipboard can be restored to a new location(s), within or outside of KULT, using the paste function.
- **Copy**: Copies highlighted text from the description tab area to the Windows® Clipboard®. The text on the clipboard can be placed at a new location(s), within or outside of KULT, using the paste function.
- **Paste**: Places text from the Windows® Clipboard® at a selected location in the description tab area.
- **Select All**: Selects everything in the description tab area.

Build tab area

The build tab area displays any error or warning messages that are generated during a code compilation or user-library build operation. When you click on a compilation-error message that is displayed in the build tab area, KULT highlights either the line of code where the error occurred or the next line, depending on how the compiler caught the error. KULT also highlights the error message. This facilitates error corrections.

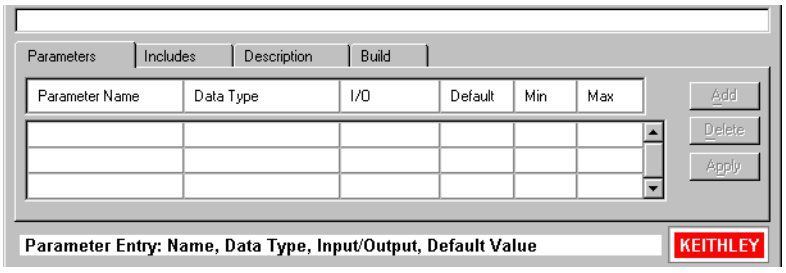
If no errors are found, the build tab area displays one of the following:

- After a compilation: **No Errors/Warnings Reported, Compilation was Successful**.
- After a build: **No Errors/Warnings Reported, Library Build was Successful**.

Understanding the status bar

The status bar, located at the bottom of the KULT dialog box, displays a description of the KULT dialog box area at the cursor location. For example, if the cursor is in the parameter tab area, the status bar describes that area, as shown in Figure 8-8.

Figure 8-8
Example of description in status bar



Understanding the menus

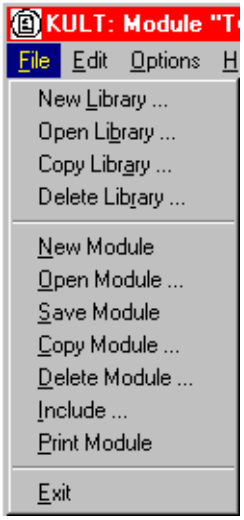
This subsection describes the menus on the menu bar, which is located at the top of the KULT dialog box.

NOTE *Most of the menu-bar menus apply to the part of the* KULT *dialog box that is above the tab areas. Refer to "Understanding the tab areas" on page 8-7 earlier in this section for information about special pop-up menus in the tab areas.*

File menu

The **File** menu is shown in Figure 8-9.

Figure 8-9
KULT File menu



File menu commands:

- **New Library**: Creates a new user library. Clicking **New Library** displays the enter library dialog box, in which you can name a new user library. Clicking **Ok** initializes and opens the new user library in place of the presently open library.

NOTE *By default, user libraries are stored in the* `C:\S4200\kiuser\usrlib` *directory. However, they can be stored on any accessible disk drive. For more information refer to the Reference Manual, Changing the active user-library directory, page 8-45.*

- **Open Library**: Opens an existing user library. Clicking **Open Library** displays the open library scroll list in which you can select an existing user library. Clicking **Ok** opens the selected library in place of the presently open library.
- **Copy Library**: Creates a copy of the currently open user library. Clicking **Copy Library** displays the enter library dialog box, in which you name the new user library into which to copy the presently open library. Thereafter, clicking **Ok** copies the presently open user library into the new library.
- **Delete Library**: Deletes an existing user library. Clicking **Delete Library** displays the delete library scroll list in which you can select the user library to be deleted. Thereafter, clicking **Ok** deletes the selected library and all of its contents.
- **New Module**: Creates a new user module. Clicking **New Module** clears module information in the KULT window and allows a new user-module name to be entered in the **Module** text box (the name must not duplicate the name of any existing user module or user library in the entire collection of user

libraries). After entering the name, clicking **Apply** initializes the new user module.

- **Open Module**: Opens an existing user module. Clicking **Open Module** displays the open module scroll list in which you can select an existing user module. Clicking **Ok** opens the selected module in place of the currently open module.
- **Save Module**: Saves the presently open user module.
- **Copy Module**: Creates a copy of the currently open user module. Clicking **Copy Module** displays the select library scroll list; there you select the user library in which to copy the presently open user module. Then, clicking **Ok** displays the enter new module dialog box; there you must enter a unique user-module name that must not duplicate the name of any existing user module or user library (in the entire collection of user libraries). Clicking **Ok** in the dialog box copies the presently open module into the selected library, under the new name (the presently open module remains open).
- **Delete Module**: Deletes a user module from the presently open user library. Clicking **Delete Module** displays the KULT: Library [`PresentlyOpenLibraryName`] list box; there you select the module to be deleted. Clicking **Ok** deletes the selected module (the presently open module continues to be displayed, even if it is the module that you deleted). However, the executable user-library file, a dynamic link library (DLL), will still contain the deleted module until you rebuild the library.
- **Include**: Imports a `*.C` file that you specify into the module code-entry area only (to insert a text or other file into the document tab area, refer to "Description tab area" on page 8-10 and read about the include pop-up menu command). Clicking **Include** displays the include other file dialog box; there you either browse and select a file or directly enter a file name and path. Thereafter, clicking **Open** inserts the file at the cursor location.

CAUTION **The File > Include command inserts *everything* from the specified file. If the specified file is the source file for a KULT user module** `<ModuleName.c>`**, everything that KULT saves into the user module (not only the C code) is imported. Hence, you must edit the entered text to remove all but the needed information. In particular, you must remove any comments of the form** `/* USRLIB MODULE ___*/`**, which KULT interprets in a special way when using the module.**

Sometimes the following is more efficient than using the File > Include command: copying only the wanted code text from the source file, then pasting it into the module code-entry area.

- **Print Module**: Prints a DOS text file containing all of the information for the presently open user module. The text file is arranged in the form that KULT uses internally.
- **Exit**: Exits KULT.

Edit menu

The **Edit** menu contains typical Windows® editing commands. (see Figure 8-10)

Figure 8-10
KULT Edit menu

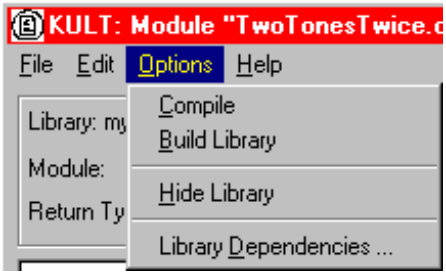
Edit menu commands:

- **Cut**: Removes highlighted text and copies it to the Windows® Clipboard®. The text on the clipboard can be restored to a new location(s), within or outside of KULT, using the paste function.
- **Copy**: Copies highlighted text to the Windows® Clipboard®. The text on the clipboard can be placed at a new location(s), within or outside of KULT, using the paste function.
- **Paste**: Places the text from the Windows® Clipboard® at a selected location.
- **Select All**: Selects everything in the module code-entry area.
- **Undo**: Allows you to reverse up to the last ten changes made in the module code-entry area.
- **Redo**: Allows you to reverse up to the last ten undo operations in the module code-entry area.

Options menu

The KULT Options menu is shown in Figure 8-11.

Figure 8-11
KULT Options menu



Options menu commands:

- **Compile**: When clicked, compiles the source files for the presently open user module into object files and checks for errors in the module.
- **Build Library:** When clicked, adds the presently open user module (or updates changes) to the presently open user library. All of the modules in the presently open user library, and any libraries on which the presently open module depends, are linked together. A dynamic link library (DLL) is then created that is accessible using UTMs in KITE.

NOTE *Some Keithley Instruments-supplied user libraries contain dependencies. If for any reason you need to build or rebuild such libraries, be sure that you specify the dependencies in the window opened by* ***Options > Library Dependencies*** *(refer to descriptions below and to details in the* *Reference Manual, Working with interdependent user modules and user libraries, page 8-51**).*
Otherwise, the Build Library function will fail. For example, `ki82ulib` *depends on* `ki590ulib` *and* `winulib`*. You must specify these dependencies before rebuilding* `ki82ulib`*, (for example, after making changes).*

- **Hide Library**: When checked, causes the current user library to be unavailable to KITE. For example, use hide library if you want to designate that a user library is only to be called by another user library and is not to be connected to a UTM.
- **Library Dependencies:** When clicked, displays the Library Dependencies list box; there you must specify each user library that is called by, and must be linked to the presently open user library. All list-box selections toggle (**Do not** hold down the control or shift key to make multiple selections).

Help menu

Help menu contains online help information about KULT:

- **Contents:** Allows access to the online KULT manual and other Model 4200-SCS reference information.
- **About KULT:** Displays the software version.

KULT Tutorials

This section includes three tutorials. Each tutorial provides step-by-step instructions for accomplishing common tasks with KULT. The name of each tutorial is included below along with a summary of topics that are discussed:

- **Tutorial 1: Creating a new user library and new user module**
  - Naming a new user library
  - Naming a new user module
  - Entering a return type
  - Entering user module code
  - Entering parameters
  - Entering header files
  - Documenting the user module
  - Saving the user module
  - Compiling the user module
  - Finding code errors
  - Building the user library to include the new user module
  - Finding build errors
  - Checking the user module
- **Tutorial 2: Creating a user module that returns data arrays**
  - Naming a new user library and new `VSweep` user module

  - Entering the `VSweep` user-module return type
  - Entering the `VSweep` user-module code
  - Entering the `VSweep` user-module parameters
  - Entering the `VSweep` user-module header files
  - Documenting the `VSweep` user module
  - Saving the `VSweep` user module
  - Compiling and building the `VSweep` user module
  - Checking the `VSweep` user module

- **Tutorial 3: Calling one user module from within another**
  - Creating the `VSweepBeep` user module by copying an existing user module
  - Calling an independent user module from the `VSweepBeep` user module
  - Specifying user library dependencies in the `VSweepBeep` user module
  - Compiling and building the `VSweepBeep` user module
  - Checking the `VSweepBeep` user module

Tutorial 1: Creating a new user library and user module

KULT is a tool that facilitates the development of user libraries. Each user library is comprised of one or more user modules, and each user module is created using the C programming language.

This subsection contains a tutorial that is designed to show you how to create a new user library and new user module. A hands-on example is provided that illustrates how to create a user library containing a user module that simply activates the internal beeper of the Model 4200-SCS.

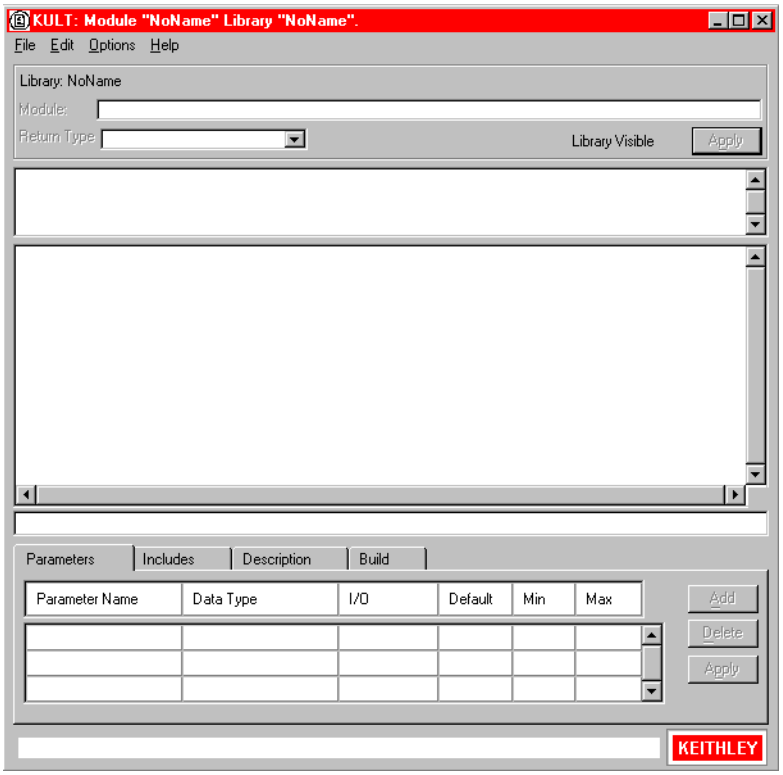
Starting KULT

To start KULT:

1. Start KULT by double-clicking the KULT icon 

on the desktop or by clicking **KULT** in the Windows® **Start** menu (**Start > Programs > Keithley > KULT**).
2. A blank KULT window appears named **KULT:** Module **"NoName"** Library **"NoName."** (see Figure 8-12)
3. Continue with Naming a new user library.

Figure 8-12
Blank KULT window

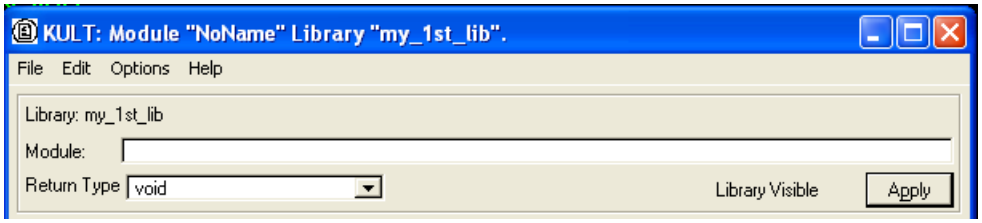


Naming a new user library

To name a new user library:

1. In the KULT file menu, click **New Library.**
2. In the Enter Library dialog box that appears, enter the new user library name. For this tutorial, enter `my_1st_lib` as the new user library name.
   The dialog box name changes to KULT**:** Module **"NoName"** Library `my_1st_lib`**,** and the name next to library in the top left side of the dialog box is now `my_1st_lib`. (see Figure 8-13)
3. Continue with Naming a new user module.

Figure 8-13
KULT window after naming user library



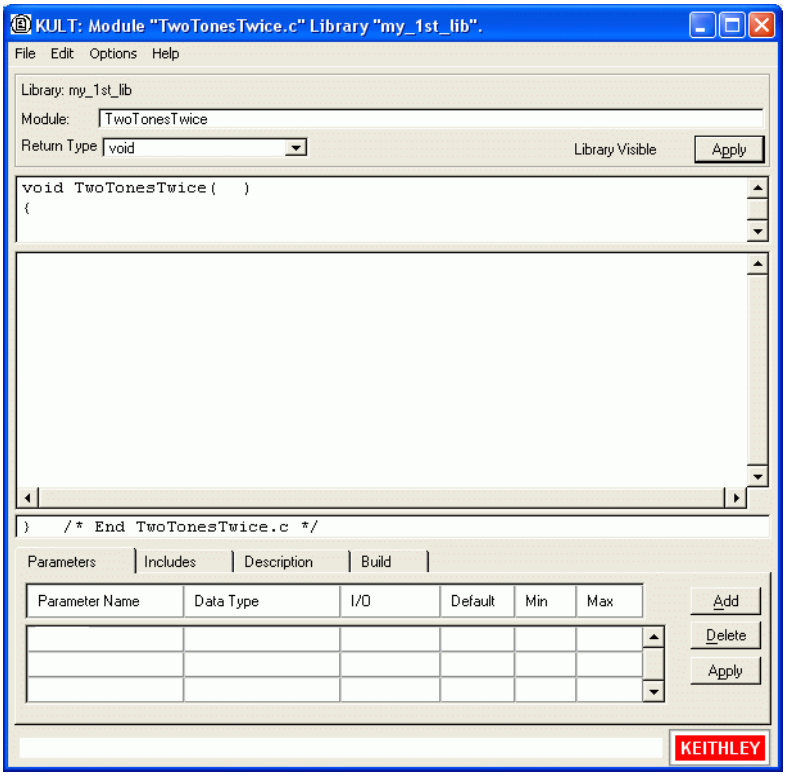
Naming a new user module

To name a new user module:

1. In the KULT file menu, click **New Module.**

2. In the module text box at the top of the KULT window, enter the new user-module name. For this tutorial, enter `TwoTonesTwice` as the new user-module name.
3. Click **Apply**.
   The KULT window changes as follows:
   - The window's name changes to **KULT:** Module **"TwoTonesTwice"** Library `my_1st_lib`.
   - You now see entries in the user-module parameters display area, in the terminating-brace display, and when you click the **Includes** tab, in the includes tab area. (see Figure 8-14)

Figure 8-14
KULT window after naming user module



NOTE *To view the entire module parameter display area, use the scroll arrows. For the* `TwoTonesTwice` *user module, the module parameter display area contains the following items:*

- `#include"keithley.h"`
- `void TwoTonesTwice ( )`

4. Continue with Entering the return type.

Entering the return type

If your user module will generate a return value, select the data type for the return value in the **Return Type** scroll box. However, the `TwoTonesTwice` user module will not produce a return value. Therefore, for the `TwoTonesTwice` module, retain the **void** default entry.

Continue with Entering user module code.

Entering user module code

Enter the C code, referring to *Reference Manual, LPT Library Function Reference, page 8-90* for a complete list of supported I/O and SMU commands.

To enter the C code:

1. Enter the new C code into the module-code entry area.
   For the `TwoTonesTwice` user module, enter the simple code listed below. The code deliberately contains a semicolon error to illustrate a KULT debug capability.

```
/* Beeps four times at two alternating user-settable frequencies. */
/* Makes use of Windows Beep (frequency, duration) function. */
/* Frequency of beep is long integer, in units of Hz. */
/* Duration of beep is long integer, in units of milliseconds. */
Beep(Freq1, 500);  /* Beep at first frequency for 500ms */
Beep(Freq2, 500);  /* Beep at second frequency */
Beep(Freq1, 500);
Beep(Freq2, 500)  /* NOTE deliberately forget semicolon initially */
```

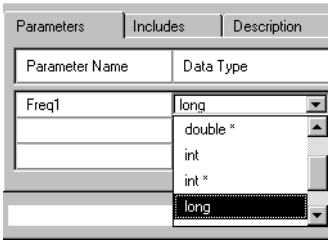
2. Continue with Entering parameters.

Entering parameters

To enter the required parameters for the code:

1. Click the **Parameters** tab (if the parameters tab area is not already displayed).
2. Click the **Add** button at the right side of the parameters tab area.
3. Under **Parameter Name**, enter the first parameter name (or accept the default). For the `TwoTonesTwice` user module, replace the default name with **Freq1**.
4. Under data type, enter the C data type for the first parameter. When you click the cell, a pop-up menu appears, displaying the allowed data types. (see Figure 8-15)

Figure 8-15
Data Type pop-up menu



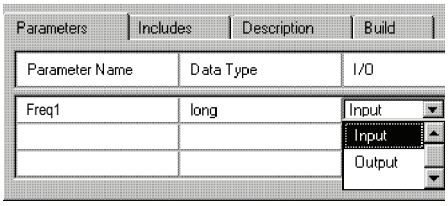
For the `TwoTonesTwice` user module, select **long** under **Data Type.**

NOTE *For an output parameter, only the following data types are acceptable: pointers (char*, float*, double*, and so on.) and arrays (*`I_ARRAY_T, F_ARRAY_T, or D_ARRAY_T`*).*

---

5. Under I/O, specify whether the first parameter is an input or output parameter. If you specified a pointer or array data type under Data Type a scroll box appears when you click the **I/O** entry cell, displaying the Input and Output options. (see Figure 8-16)

Figure 8-16
I/O pop-up menu for pointers and arrays



If you do not specify a pointer or array data type under **Data Type**, you cannot change the default **Input** entry. For the `TwoTonesTwice` user module, the default **Input** entry is correct.

6. Under **Default**, **Minimum**, and **Maximum**, enter default, minimum, and maximum values for the parameter: to simplify and limit the choices to the user. For the `TwoTonesTwice` user module, enter **1000**, **800**, and **1200**, respectively.
7. Repeat steps 2 through 6 for all additional input and output parameters for the user module that you are creating. For the `TwoTonesTwice` module, add one more parameter, as shown in Table 8-1.

Table 8-1
TwoTonesTwice entries for second line of Parameters tab area

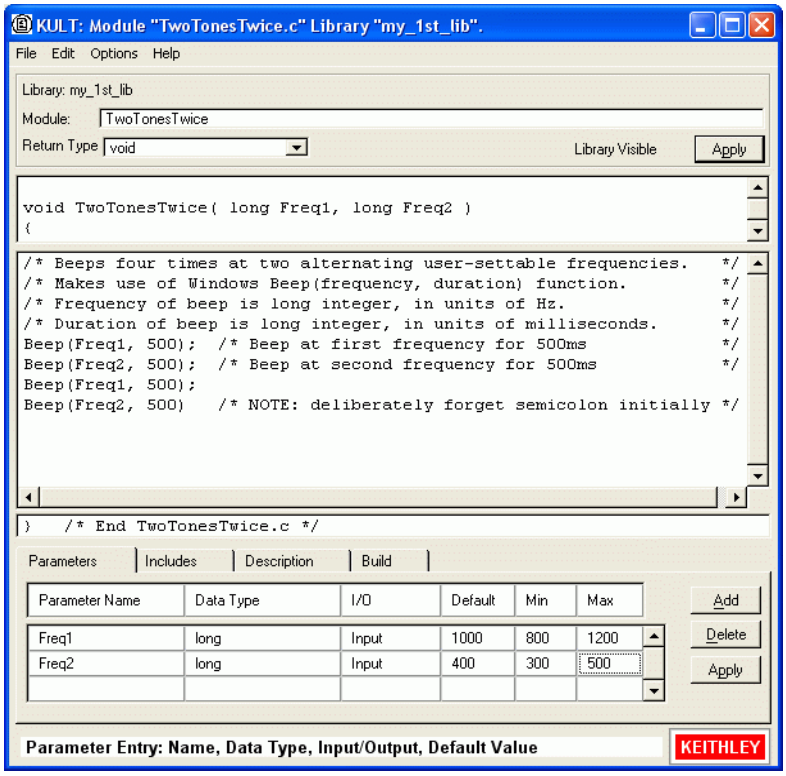
Parameter name	Data type	I/O	Default	Min	Max
Freq2	long	Input	400	300	500

8. Click **Apply.**

NOTE *The two **Apply** buttons, at the top and bottom of the dialog box, act identically.*

In the module-parameter display area, the function prototype now includes the declared parameters. (see Figure 8-17)

Figure 8-17
KULT window after entering and applying code and parameters



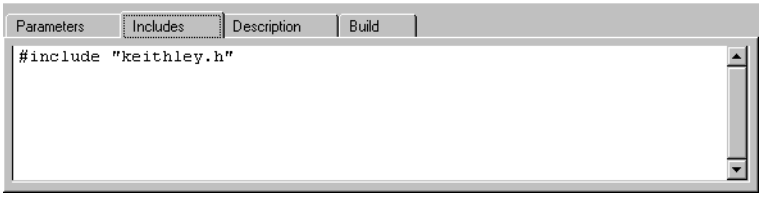
9. Continue with Entering header files.

Entering header files

To enter the header files:

1. Click on the **Includes** tab at the bottom of the dialog box. The Includes tab area appears. (see Figure 8-18)

Figure 8-18
Default Includes tab area



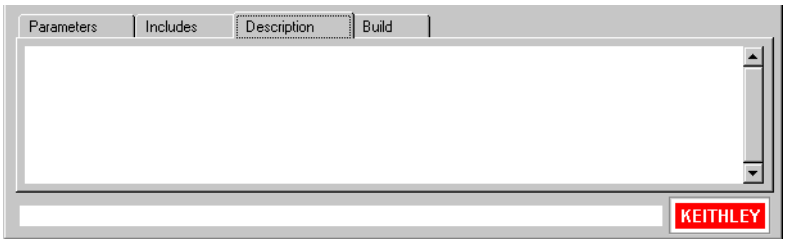
2. Enter any additional header files that are needed by the user module. No additional header files are needed for the `TwoTonesTwice` user module or for any of the user libraries supplied by Keithley Instruments.
3. Click **Apply**.
4. Continue with Documenting the user module.

Documenting the user module

To document the user module:

1. Click the **Description** tab at the bottom of the dialog box. The description tab area opens. (see Figure 8-19)

Figure 8-19
Description tab area



2. Enter any text needed to adequately document the user module to the KITE user, who does not see the comments that you include with the code.

CAUTION **Do not use C-code comment designators (/*, */, or //) in the Description tab area. When the user module is compiled, KULT also evaluates the Description text. C-code comment designators in the Description tab area can be misinterpreted, causing errors.**

NOTE ***Do not*** *place a period in the first column (the left-most position) of any line in the* description *tab area. Any text after a first-column period will not be displayed in the UTM description area.*

For the `TwoTonesTwice` user module, enter the following information in the Description tab area:

MODULE:
`TwoTonesTwice`

DESCRIPTION:
Execution results in sounding of four beeps at two alternating user-settable frequencies. Each beeps sounds for 500ms.

INPUTS:
`Freq1 (double)` is the frequency, in Hz, of the first and third beep.
`Freq2 (double)` is the frequency, in Hz, of the second and fourth beep.

OUTPUTS:
None

RETURN VALUES:
None

3. Continue with Saving the user module.

Saving the user module

Save the user module by clicking **Save Module** in the **File** menu.

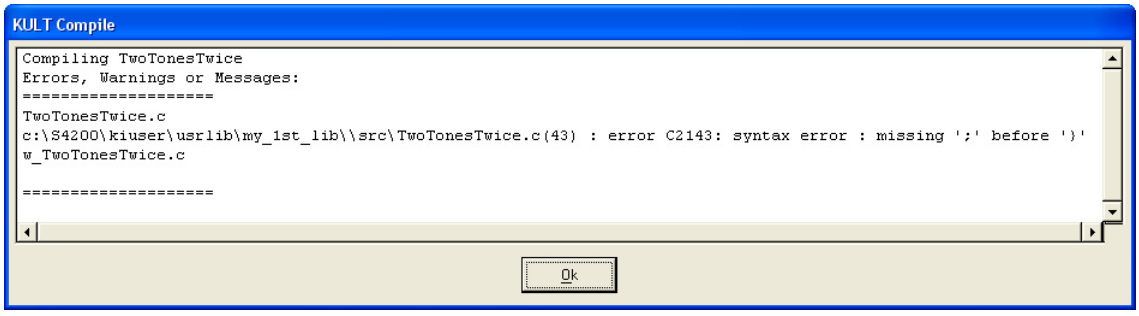
Continue with Compiling the user module.

Compiling the user module

To compile the user module:

1. Click the **Build** tab at the bottom of the dialog box. The build tab area opens. After you compile a user module, the build tab area displays either a confirmation that the module compiled successfully or displays one or more compile-error messages.
2. In the **Options** menu, click **Compile**. The following occurs:
   a. The user-module C source-code file is compiled.
   b. The KULT Compile message box indicates the compilation progress and, if problems are encountered, displays error messages. For example, when you first compile the `TwoTonesTwice` user module (with a missing semicolon), you see the KULT Compile message box shown in Figure 8-20.

Figure 8-20
KULT Compile message box with error message



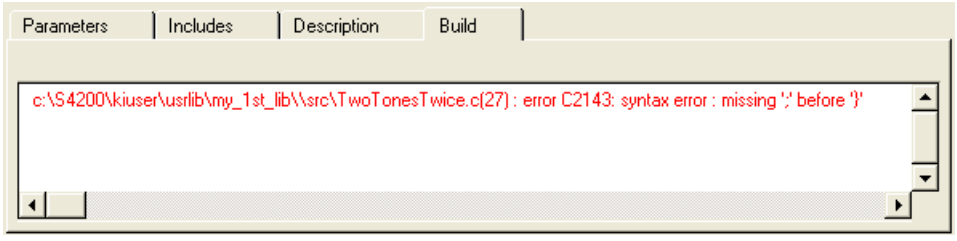
   c. When the KULT Compile message box closes (or, if there are error messages, when you click **Ok**), the build tab area displays either of the following:
      - If the compilation was successful, the following message appears: **No Errors/ Warnings Reported, Compilation was Successful.**
      - If the compilation was unsuccessful, the error message(s), if any, that was displayed in the KULT Compile message box also displays in the **Build** tab area.

NOTE *True compilation errors (errors that prevent the user module from compiling) are displayed in red.*

Warnings, which disclose suspect code that does not prevent compilation (such as an unused variable declaration) are displayed in blue.

   - For example, after you first compile the `TwoTonesTwice` user module (with the semicolon error) and click **Ok** in the KULT Compile message box, the build tab area appears as in Figure 8-21.

Figure 8-21
Compile error message in the Build tab area



3. Continue with Finding code errors.

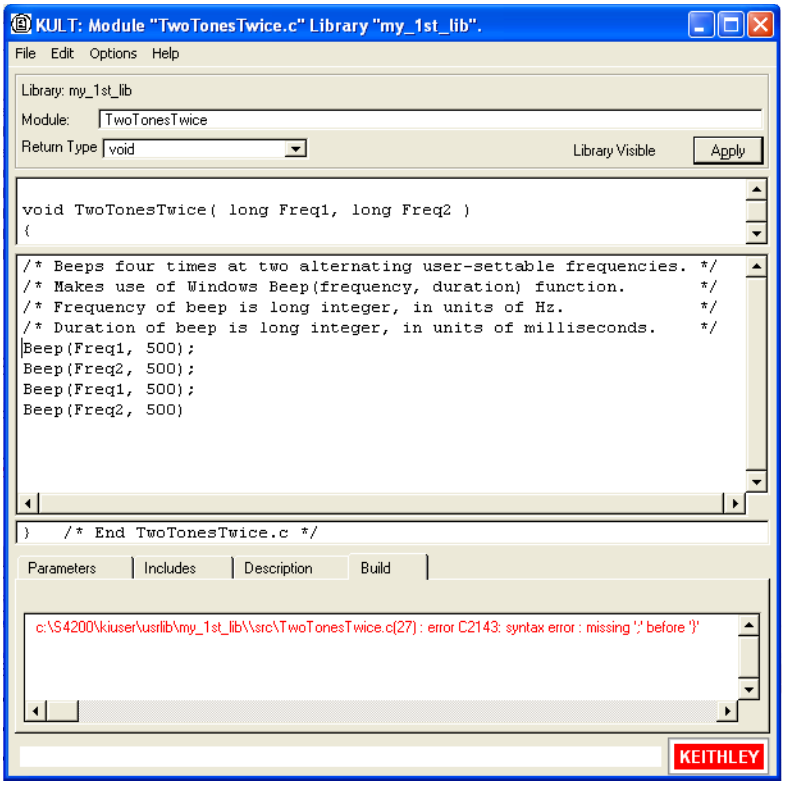
Finding code errors

When you click on a compilation-error message that is displayed in the build tab area, KULT highlights either the line of code where the error occurred or the next line, depending on how the compiler caught the error. KULT also highlights the error message.

To find code errors for the `TwoTonesTwice` user module:

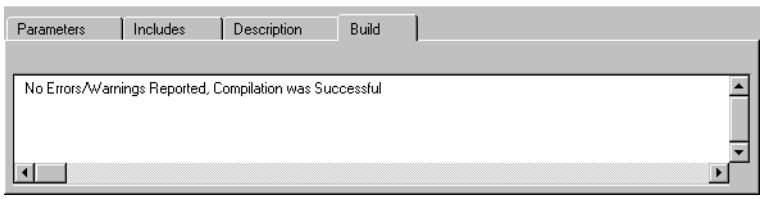
1. Click the error message. The last line of code (the line missing the semicolon) is highlighted, as shown in Figure 8-22.

Figure 8-22
Finding a code error



2. Add the missing semicolon at the end of the code [`Beep(Freq2,500);`], and delete the comment about the missing semicolon.
3. Save the user module.
4. Compile the user module again.
   - The KULT Compile message box should now display no error messages and disappears automatically.
   - The build tab area should display the successful-compilation message. (see Figure 8-23)

Figure 8-23
Successful-compilation message displayed in build tab area



5. Continue with Building the user library to include the new user module.

Building the user library to include the new user module

After you have successfully compiled the user module, build the user library (or rebuild the user library) to include the module.

To build the user library:

1. Keep the Build tab area open.
2. In the **Options** menu, click **Build**. The following occurs:
   a. The user library builds. All of the user modules in the presently open user library, and any libraries on which the presently open user module depends, are linked together. A DLL is then created that is accessible using UTMs in KITE.
   b. The KULT Build Library message box (similar to the KULT Compile message box) indicates the build progress and, if linker problems are encountered, displays error messages (when you build the `TwoTonesTwice` user module, you should see no errors).
   c. When the KULT Build Library message box closes (or, if there are error messages when you click **Ok**), the Build tab area displays either of the following:
      - If the compilation was successful, the following message appears: **No Errors/ Warnings Reported, Library Build was Successful.**
      - If the compilation was unsuccessful, error messages, if any, that were displayed in the KULT Build Library message dialog box also display in the Build tab area (in red, only).

Finding build errors

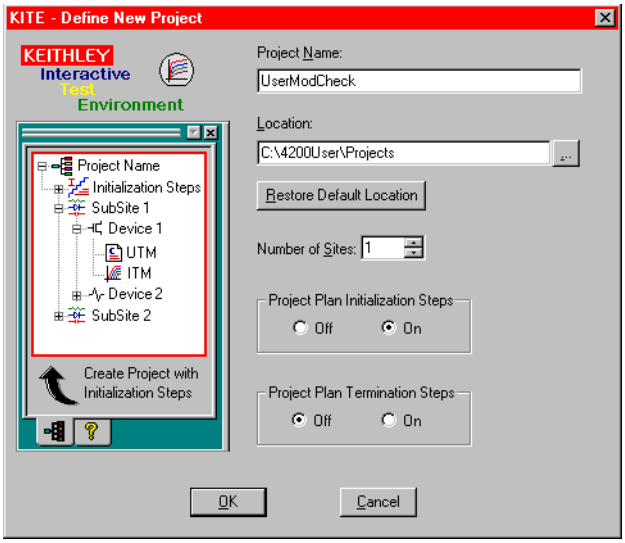
Find build errors using the information in the error message.

Checking the user module

To check the user module by creating and executing a UTM in KITE:

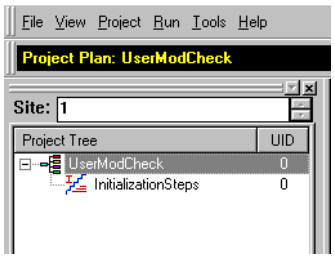
1. Create a simple KITE project to check user modules, as follows:
   a. Start KITE by double-clicking the KITE icon on the desktop or by clicking **KITE** in the Windows® start menu (**Start > Programs > Keithley > KITE**).
   b. In the KITE **File** menu, click **New Project.** The KITE - Define New Project dialog box appears.
   c. In the KITE - Define New Project dialog box, do the following:
      - In the Project Name text box, enter a project name. A logical choice is **UserModCheck**. For this tutorial, enter **UserModCheck**.
      - Under Project Plan Initialization Steps, click **On**. (see Figure 8-24)

Figure 8-24
Defining the UserModCheck project



d. In the KITE - Define New Project dialog box, click **Ok**. The plan for the new project appears in the project navigator. (see Figure 8-25)

Figure 8-25
Initial UserModCheck project



2. Insert a new UTM in the project navigator. This will be used to execute the user module that you wish to check.

NOTE *For simplicity, user modules that you create with* KULT *can be checked by creating Initialization Steps UTMs in the project navigator. UTMs can also be attached to specific devices on specific subsites. This tutorial uses the Initialization Steps approach.*

Do the following in KITE:

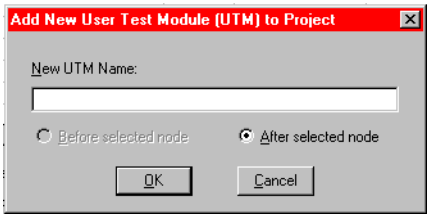
a. In the Check Project Navigator, single-click on **Initialization Steps**.

b. Single-click on the **Add new UTM** icon, found at the top of the KITE dialog box, or select **Project > New UTM**.

The Add New User Test Module (UTM) to Project dialog box appears. (see Figure 8-26)

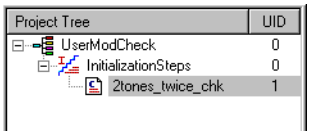
The Add New User Test Module (UTM) to Project dialog box appears. (see Figure 8-26)

Figure 8-26
Add New User Test Module (UTM) to Project dialog box



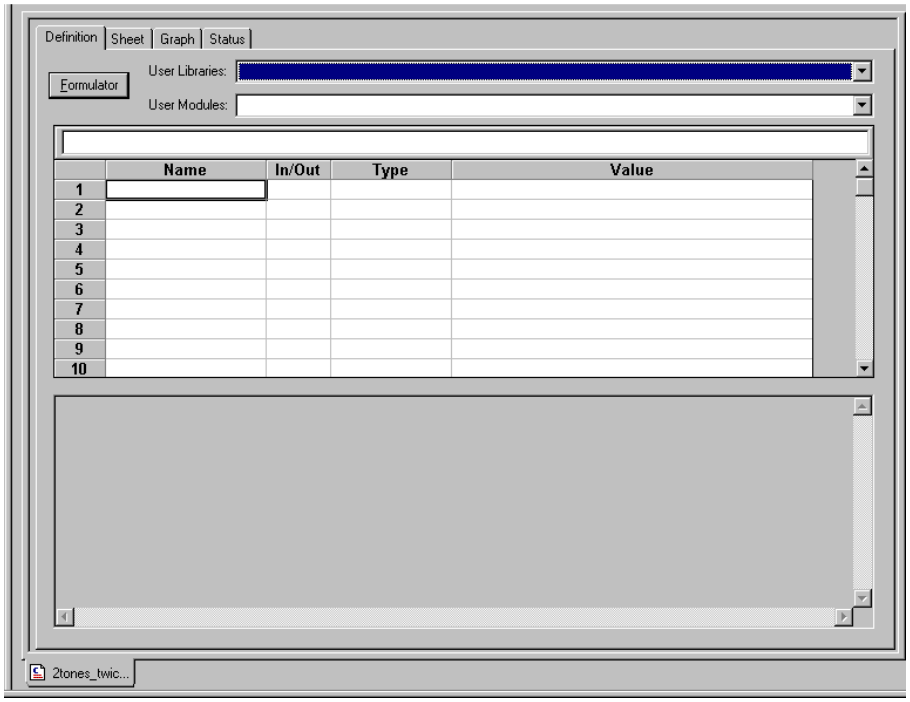
c. In the Add New User Test Module (UTM) to Project dialog box, enter the name for the UTM. For the `TwoTonesTwice` user module, enter `2tones_twice_chk`.
d. Click **Ok**. The new UTM now appears under Initialization Steps in the project navigator. (see Figure 8-27)

Figure 8-27
New UTM inserted in the project plan



3. Configure the new UTM so that it executes the user module that you wish to check. Do the following:
   a. In the project navigator, double-click the name of the new UTM (for the tutorial, double-click `2tones-twice-chk`). A blank UTM Definition document appears. (see Figure 8-28)

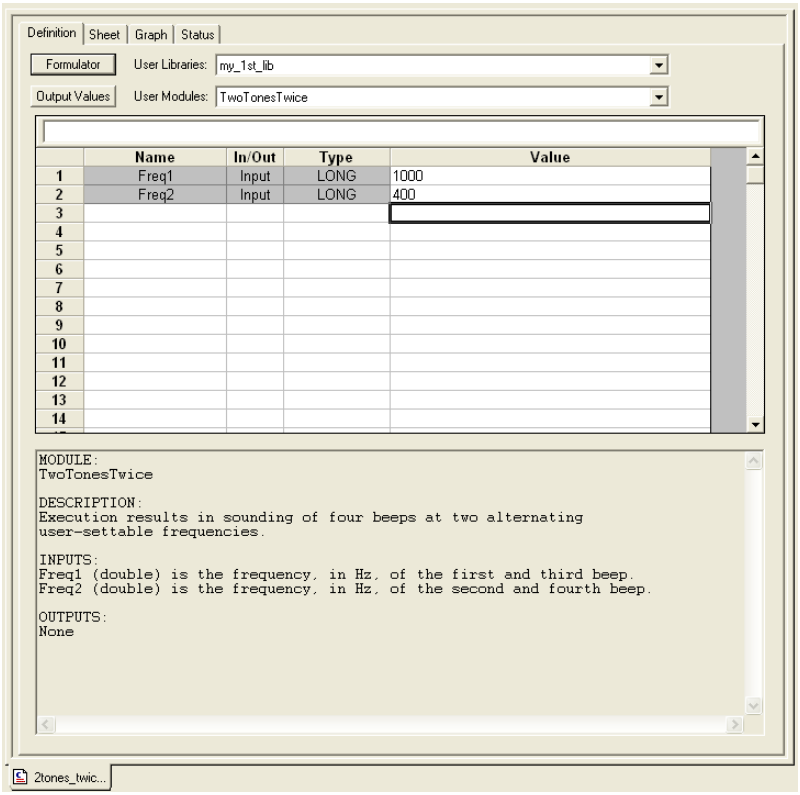
Figure 8-28
Blank UTM Definition document



b. In the User Libraries scroll box of the UTM Definition document, select the user library that contains the user module that you wish to test. For the tutorial, select `my_1st_lib`.

c. In the User Modules scroll box of the UTM Definition document, select the user module that you wish to test (for the tutorial, select `TwoTonesTwice`). The UTM now displays the configuration parameters for the selected user module. (see Figure 8-29)

Figure 8-29
Configured UTM



d. Accept the default parameters for now (you can experiment later after you establish that the user module executes correctly).

e. Save the UTM and the project by clicking the **Save All** icon at the top of the KITE screen, or by clicking **Save All** in the KITE **File** menu.

f. Execute the UTM by clicking the green **Run** test/plan icon at the top of the KITE screen, or by selecting **Run** in the KITE run menu.

The Run test/plan icon becomes gray, the test executes, and the square Abort test/plan icon illuminates red for the duration of the test .

NOTE *If you need to abort the test during execution, click the red* ***Abort*** *Test/Plan icon.*

When you execute the `TwoTonesTwice` user module in the `2tones_twice_chk` UTM, you should hear a sequence of four tones, sounded at alternating frequencies.

4. If the user module that you created generates data, check the execution results in the UTM Data worksheet. To view the Data worksheet, click the **Sheet** tab at the top of the UTM Definition document.

NOTE *For a UTM, the Data worksheet (and, if defined, the Graph document) always update at the conclusion of execution. Remember, you cannot view numerical and graphical results in real time.*

The `TwoTonesTwice` user module, executing in the `2tones_twice_chk` UTM, generates no data. For an example of numerical data, see the Tutorial 2 data under "Checking the VSweep user module" on page 8-33 later in this section.

For more details on building a project, creating a UTM, and executing a UTM, refer to the following subsections in the Reference Manual, Keithley Interactive Test Environment (KITE), page 6-1.

- Building, modifying, and deleting a Project Plan
- Configuring the UTMs
- Executing Project Plans, Subsite Plans, Device Plans, and tests in Section 6 of the Reference Manual

Tutorial 2: Creating a user module that returns data arrays

This subsection provides a tutorial that is designed to help you to use array variables in KULT. It also illustrates the use of return types (or codes), and the use of two functions from the Keithley Linear Parametric Test Library (LPTLib).

Most of the basic steps that were detailed above are only abbreviated in this tutorial. Before doing the following tutorial, we suggest first completing "Tutorial 1: Creating a new user library and user module" on page 8-16 explained earlier in this section.

Naming new user library and new VSweep user module

To name new user library and new VSweep user module:

1. Start KULT by double-clicking the **KULT** icon on the desktop.
2. In the KULT file menu, click **New Library.**
3. In the Enter Library dialog box that appears, enter `my_2nd_lib` as the new user library name.
4. In the KULT file menu, click **New Module.**
5. In the **Module** text box at the top of the KULT dialog box, enter **VSweep** as the new module name.
6. Click **Apply**.
7. Continue with Entering the VSweep user-module return type.

Entering the VSweep user-module return type

The VSweep user module generates an integer return value. Therefore, select **int** in the Return Type scroll box.

Continue with Entering the VSweep user-module code.

Entering the VSweep user-module code

In the module code-entry area, enter the C code for the VSweep user module. The code is listed below (enter the code with KULT dialog box in full screen view).

```
/* VSweep module
   --------------
Sweeps through specified V range & measures I, using specified number of
points.
```

```
Places forced voltage & measured current values (Vforce and Imeas) in
output arrays.
```

```
NOTE For n increments, specify n+1 array size (for both NumIPoints and
NumVPoints).
```

```
*/
double vstep, v;/* Declaration of module internal variables. */
int i;
if ( (Vstart == Vstop) )/* Stops execution and returns -1 if */
    return( -1 );/* sweep range is zero.              */

if ( (NumIPoints != NumVPoints) )/* Stops execution and returns -2 if */
    return( -2 );/* V and I array sizes do not match.  */

vstep = (Vstop-Vstart) / (NumVPoints -1);/* Calculates V-increment size.
*/

for(i=0, v = Vstart; i < NumIPoints; i++)/* Loops through specified num-
ber of */
   /* data points. */
{
forcev(SMU1, v);/* LPTLib function forceX, which forces a V or I.  */
measi(SMU1, &Imeas[i]);/* LPTLib function measX, which measures a V or
I. */
   /* Be sure to specify the *address* of the array. */

Vforce[i] = v;/* Returns Vforce array for display in UTM Sheet. */

v = v + vstep;/* Increments the forced voltage. */
 }

return( 0 );/* Returns zero if execution Ok.*/
```

Continue with Entering the VSweep user-module parameters.

Entering the VSweep user-module parameters

To enter the required parameters for the code:

1. Click the **Parameters** tab (if the Parameters tab area is not already displayed).
2. Enter the information for the two voltage input parameters, as shown in Table 8-2. Click the **Add** button before adding each new parameter.

Table 8-2
VSweep entries for the two voltage input parameters

Parameter name	Data type	I/O	Default	Min	Max
Vstart	double	Input	0	-200	200
Vstop	double	Input	5	-200	200

NOTE *When executing the Vsweep user module in a UTM, the start and stop voltages (Vstart and Vstop) must differ. Otherwise, the first return statement in the code halts execution and returns an error number (-1). When a user module is executed using a*

KITE *UTM, this return code is stored in the UTM Data worksheet. The return code is stored in a column that is labeled with the user-module name.*

3. Click **Add**, and on the third line, enter the measured-current parameter information shown in Table 8-3.

Table 8-3
Entries for the VSweep measured-current parameter

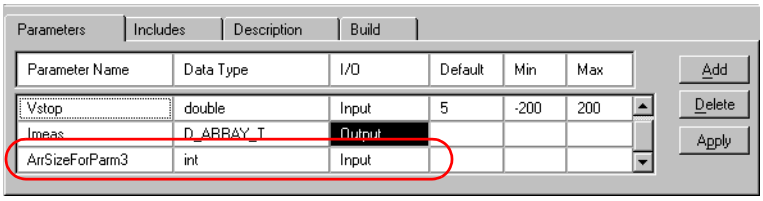
Parameter name	Data type	I/O	Default	Min	Max
Imeas	`D_ARRAY_T`	Output			

NOTE *The following about the double-precision* `D_ARRAY_T array` *type, which applies also to the* `I_ARRAY_T and F_ARRAY_T` *array types:*

- `D_ARRAY_T, I_ARRAY_T and F_ARRAY_T` *are special array types that are unique to KULT.*
- *You cannot enter values in the Default, Min, and Max fields.*
- *On the scroll bar in the Parameters tab area, there is a space below the slider. This space indicates the existence of a hidden fourth line of incomplete parameter information: the array-size parameter specification (described in the next step).*

4. Scroll down to reveal line four of the Parameters tab area. Line four contains the KULT entered array-size parameter for the array that is specified on line three. (see Figure 8-30)

Figure 8-30
KULT-entered array-size parameters



NOTE *The following about the array size specification line:*

- *KULT enters initial information on this line automatically.*
- *The default Parameter Name entry is only a description of the required array-size parameter. You must replace it with an appropriate array-size parameter (for the* `VSweep` *user module, the correct entry is NumIPoints, as required per the user module code).*
- *The Data Type and I/O entries are correct as entered.*
- *You can enter a Default, Min, and Max array size.*
- *An array-size parameter line always appears after an array parameter line. You must always enter array-size parameters in the specified line.*

5. In the fourth line, under Parameter Name, in place of the designation ArrSizeForParm3, enter the parameter NumIPoints.
6. On the fourth line, under Default, enter the number **11** for the default current-array size.

7. Click **Add** and, on the 5th line, enter the forced-voltage parameter information shown in Table 8-4.

Table 8-4
Entries for the VSweep forced-voltage parameter

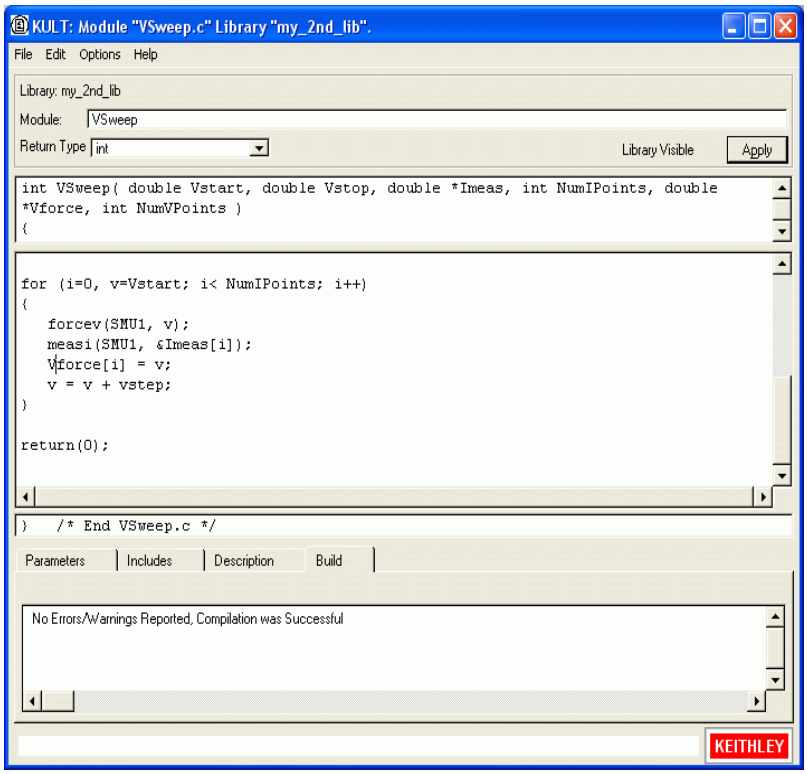
Parameter name	Data type	I/O	Default	Min	Max
Vforce	D_ARRAY_T	Output			

8. In the sixth line, under **Parameter Name**, in place of the designation `ArrSizeForParm5`, enter `NumVPoints`.
9. In the sixth line, under **Default**, enter the number **11** for the default voltage array size.

NOTE *When executing the* `VSweep` *user module in a UTM, the current and voltage array sizes must match;* `NumIPoints` *must equal* `NumVPoints`*. Otherwise, the second* `return` *statement in the code halts execution and returns an error number (-2) in the* `VSweep` *column of the UTM Data worksheet.*

10. Click **Apply.** In the module-parameter display area, the function prototype now includes the declared parameters. (see Figure 8-31)

Figure 8-31
VSweep user-module dialog box after entering and applying code and parameters



11. Continue with Entering the `VSweep` user-module header files.

Entering the VSweep user-module header files

You do not need to enter any header files for the `VSweep` user module. The default `keithley.h` header file is sufficient. Continue with Documenting the VSweep user module.

Documenting the VSweep user module

After clicking the Description tab, enter documentation for the user module, based on the comments provided in the code and other information about the module. Then continue with Saving the VSweep user module.

Saving the VSweep user module

Save the user module by clicking **Save Module** in the **File** menu. Then continue with Compiling and building the VSweep user module.

Compiling and building the VSweep user module

To compile the user module:

1. Click the **Build** tab at the bottom of the dialog box. The Build tab area opens.
2. In the Options menu, click **Compile**. The user module compiles. If the code and parameters were entered as specified, you should not see error messages.

NOTE *If you do see error messages, check for typographical errors; then fix and recompile the user module. If necessary, review "Finding code errors" on page 8-24 earlier in this section.*

---

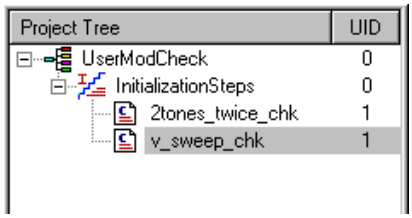
3. In the **Options** menu, click **Build**. The user library builds. You should not see error messages.
4. Continue with Checking the VSweep user module.

Checking the VSweep user module

Check the user module by creating and executing a UTM in KITE, using the general procedure described in Tutorial 1 under "Checking the user module" on page 8-25 earlier in this section. Before proceeding, observe the following guidelines:

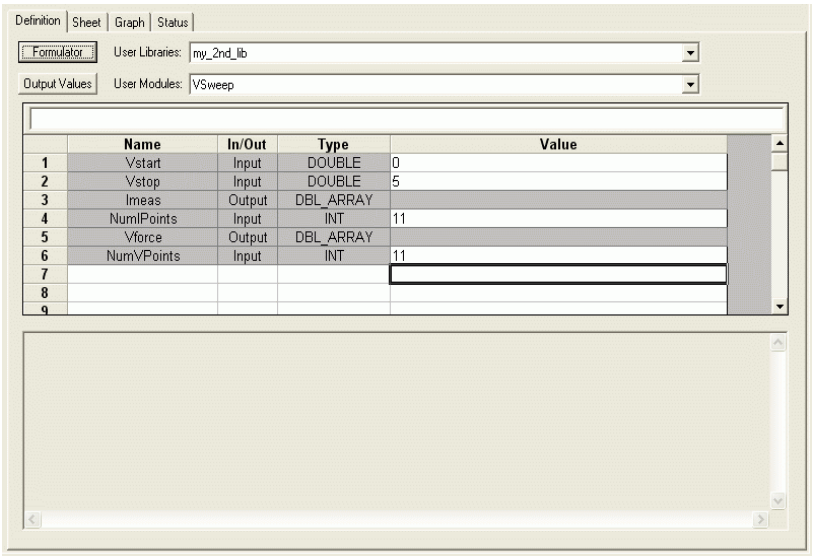
1. Connect a 1 k Ω resistor between the FORCE terminal of the ground unit (GNDU) and the FORCE terminal of SMU1.
2. Instead of creating a new project, reuse the UserModCheck project that you created in Tutorial 1. Add a new UTM called `v_sweep_chk`. You will subsequently use `v_sweep_chk` to execute the VSweep user module. (see Figure 8-32)

Figure 8-32
New v_sweep_chk check UTM inserted in the project plan



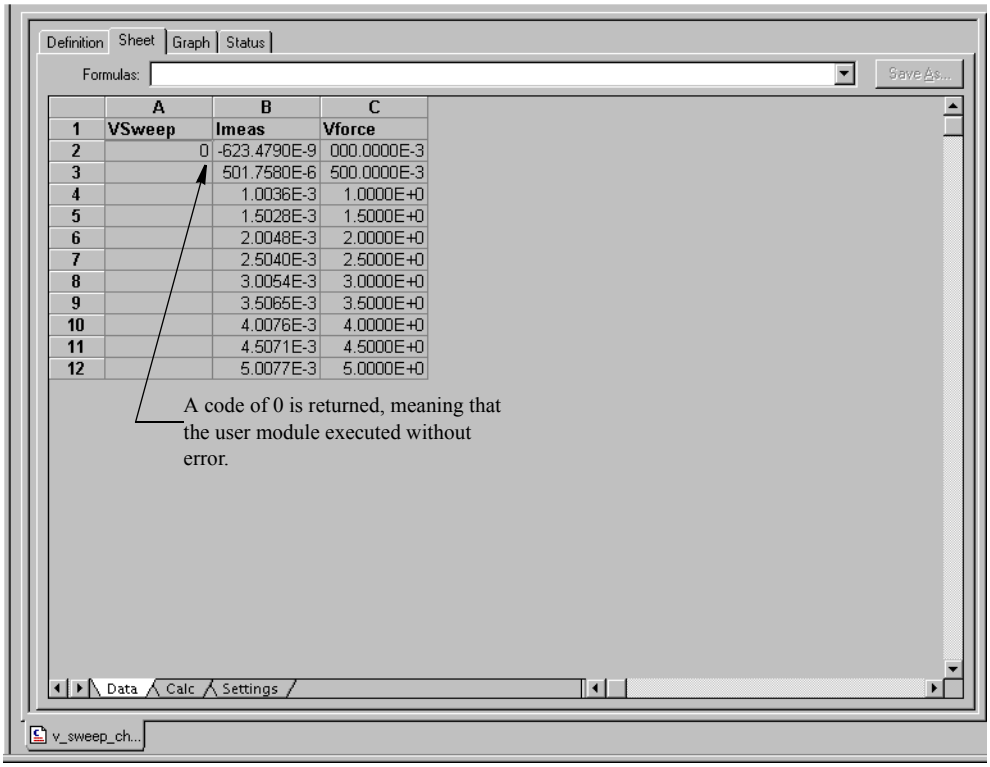
3. Configure the `v_sweep_chk` UTM to execute the VSweep user module, which is found in the `my_2nd_lib` user library. Figure 8-33 shows the UTM configured with the default parameter values.

Figure 8-33
Configure v_sweep_chk UTM



4. Execute the UTM as directed in Tutorial 1, using the default parameter values.
5. At the conclusion of execution, review the results in the Data worksheet. If you connected a 1 k Ω resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, the results should appear similar to the results in Figure 8-34. The current/voltage ratio for each row of results should be approximately 1 mA / V.

Figure 8-34
Reviewing Data worksheet after executing a UTM



Tutorial 3: Calling one user module from within another

KULT allows a user module to call other user modules. A called user module may be located within the same user library as the calling module or may be located in another user library. This subsection provides a brief tutorial that illustrates application of such dependencies. It also illustrates the **File > Copy Module** command.

In this tutorial, you create a new user module using two user modules that were created in the previous tutorials: "Tutorial 1: Creating a new user library and user module" on page 8-16, and "Tutorial 2: Creating a user module that returns data arrays" on page 8-29 earlier in this section:

- The VSweep user module, in the `my_2nd_lib` user library (a copy of which will be used as the dependent user library).
- The `TwoTonesTwice` user module, in the `my_1st_lib` user library, which is the independent user library that will be called by the VSweep user module.

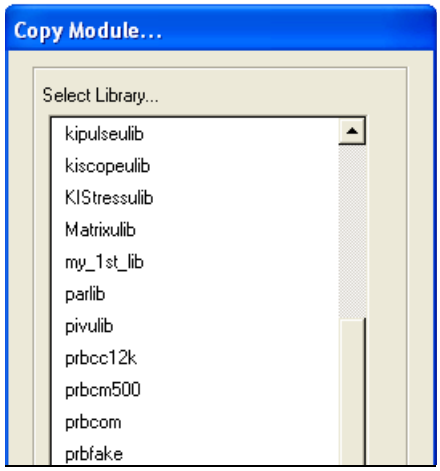
A copy of the VSweep user module, to be called `VSweepBeep`, calls the `TwoTonesTwice` user module to signal the end of execution.

Creating VSweepBeep user module copying existing user module

To create the new user module:

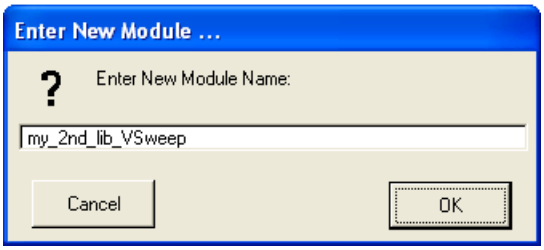
1. Start KULT.
2. Open the VSweep user module as follows:
   a. Open `my_2nd_lib` by:
      i. Clicking **File > Open Library**
      ii. Selecting `my_2nd_lib` from the list box that appears
      iii. Clicking **Ok**.
   b. Click **File > Open Module**, select `VSweep.c` from the list box that appears, and click **Ok**.
3. Copy `VSweep.c` to the new user module, `VSweepBeep`, as follows:
   a. In the **File** menu, click **Copy Module**. The Copy Module list box appears. (see Figure 8-35)

Figure 8-35
Copy Module list box



   b. In the Copy Module list box, select `my_2nd_lib` (in this specific case, the user library for the copy is the same as the user library for the source) and click **Ok**. The Enter New Module dialog box appears. (see Figure 8-36)

Figure 8-36
Enter New Module dialog box



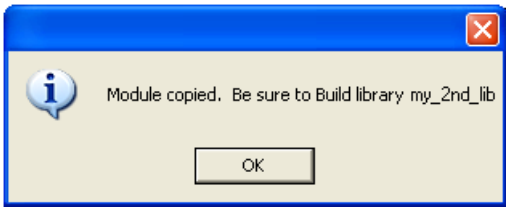
c. In the Enter New Module dialog box, enter the name **VSweepBeep** instead of the default name, and click **Ok**.

NOTE *Each user module must have a unique name, regardless of the user library in which it resides. That is, in the presently active user library folder, there can only be one user module of a given name in its entire collection of libraries*

More than one collection of user libraries can be maintained and accessed, each collection residing in a separate usrlib. However, only one usrlib can be active at a time. For more information, refer to the Reference Manual, Managing user libraries, page 8-40 .

KULT creates a copy of the user module under the new name and displays a message indicating the need to rebuild the user library. (see Figure 8-37)

Figure 8-37
Library build message box



You may skip the rebuild for now. Continue with the next step.

4. Open the new `VSweepBeep` user module.
   a. Click **File > Open Module.**
   b. Select **VSweepBeep.c** from the list box that appears. The KULT dialog box displays the `VSweepBeep` user module.
5. Continue with Calling independent user module from VSweepBeep user module.

NOTE *You can also create a copy of the presently open user module in the same user library as follows:*

1. *Enter a new name in the User Module text box.*
2. *Click Apply. Before using the user module, you must save and compile it, and then rebuild the user library.*

Calling independent user module from VSweepBeep user module

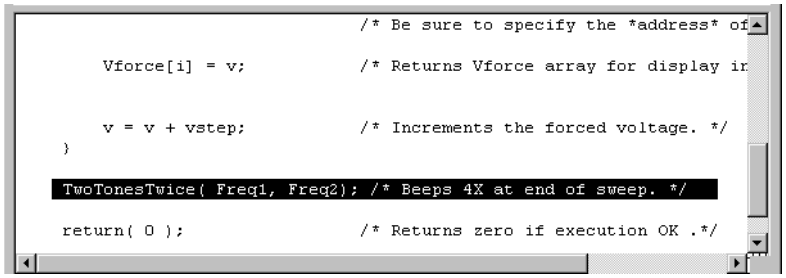
To call the `TwoTonesTwice` user module at the end of the `VSweepBeep` user module:

1. At the end of `VSweepBeep`, and just before the `return(0)` statement, add the following statement:

   ```
   TwoTonesTwice(Freq1, Freq2); /* Beeps 4X at end of sweep. */
   ```

   (see Figure 8-38)

Figure 8-38
Calling TwoTonesTwice from VSweepBeep



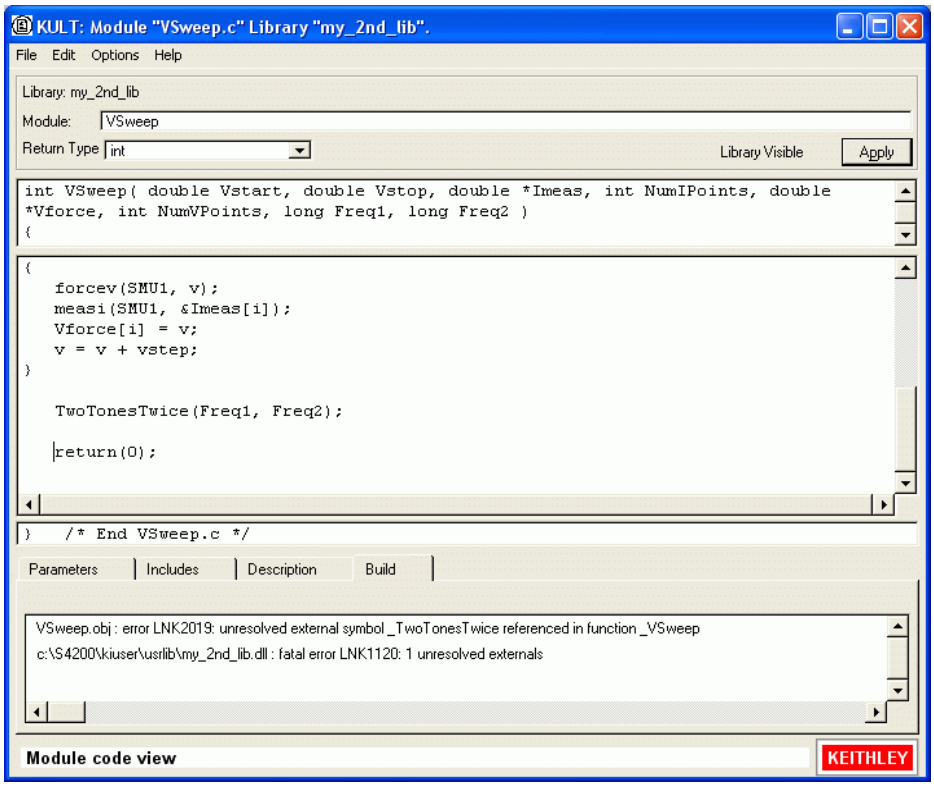
2. In the Parameters tab area, add the **Freq1** and **Freq2** parameters, just as you did when you created the `TwoTonesTwice` user module, changing the Default, Min, and Max values if you prefer. (see Table 8-5)

Table 8-5
Parameter entries for the called user module, TwoTonesTwice

Parameter name	Data type	I/O	Default	Min	Max
Freq1	long	Input	1000	800	1200
Freq2	long	Input	400	300	500

3. Click **Apply**. The **Freq1** and **Freq2** parameters are added to the function prototype. (see Figure 8-39)

Figure 8-39
Completed VSweepBeep user module



4. Continue with Specifying user library dependencies in VSweepBeep user module.

Specifying user library dependencies in VSweepBeep user module

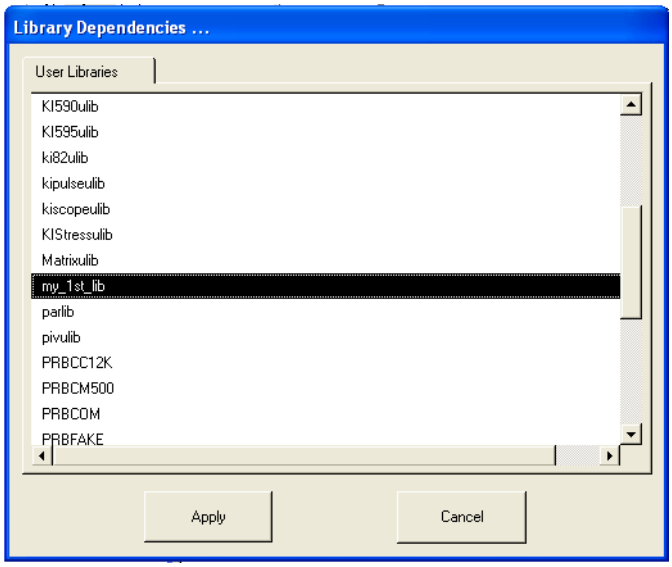
Before compiling the presently open user module, you must specify all user libraries on which the user module depends (that is, the other user libraries containing user modules that are called).

The `VSweepBeep` user module depends on the `my_1st_lib` user library.

To specify this dependency:

1. In the **Options** menu, click **Library Dependencies**. The Library Dependencies list box appears. (see Figure 8-40)

Figure 8-40
Library Dependencies list box



2. In general, in the Library Dependencies list box, select all user libraries on which the presently open user module depends (each selection toggles on and off). For the `VSweepBeep` module, only select `my_1st_lib`. (see Figure 8-40)
3. Click **Apply**.
4. Continue with Compiling and building the VSweepBeep user module.

Compiling and building the VSweepBeep user module

To compile and build the `VSweepBeep` user module:

1. Save the `VSweepBeep` user module.
2. Click the **Build** tab at the bottom of the dialog box. The Build tab area opens.
3. In the **Options** menu, click **Compile**. The user module compiles. If the code and parameters were entered as specified, you should not see error messages.

NOTE *If you do see error messages, check for typographical errors; then fix and recompile the module. If necessary, review "Finding code errors" on page 8-24 earlier in this section.*

---

4. In the **Options** menu, click **Build**. The user library builds. You should not see error messages.
5. Continue with Checking the VSweepBeep user module.

Checking the VSweepBeep user module

Check the user module just as you did in Tutorials 1 and 2, by creating and executing a UTM in KITE (refer to the general procedure described in Tutorial 1 under "Checking the user module" on page 8-25).

The text of the tutorial-specific guidelines below are almost identical to the text of the Tutorial 2 guidelines. Also, the data produced should be the same as the Tutorial 2 data. However, additionally, four beeps should sound at the end of execution, just as when you tested the `TwoTonesTwice` user module in Tutorial 1.

Before proceeding, observe the following guidelines:

1. Connect a 1 k Ω resistor between the FORCE terminal of the GNDU and the FORCE terminal of SMU1.
2. Instead of creating a new project, reuse the UserModCheck project that you created in Tutorial 1. Add to this project a UTM called `v_sweep_bp_chk`.
3. Configure the `v_sweep_bp_chk` UTM to execute the `VSweepBeep` user module, which is found in the `my_2nd_lib` user library.
4. Execute the `v_sweep_bp_chk` UTM. Near the end of a successful execution, you should hear a sequence of four tones, sounded at alternating frequencies.
5. At the conclusion of execution, review the results in the Data worksheet (or the Graph document, if configured). If you connected a 1 k Ω resistor between SMU1 and GNDU, used the default UTM parameter values, and executed the UTM successfully, your results should appear similar to the results in Figure 8-34 at the end of Tutorial 2. The current/voltage ratio for each row of results should be approximately 1 mA / V.

Advanced KULT features

The following text discusses the advanced features of KULT in the following sections:

- Managing user libraries
- Working with interdependent user modules and user libraries
- Understanding user module locking
- Debugging user modules using Microsoft Visual C++

NOTE *The Microsoft Visual Studio C++ compiler is used to create, modify, and debug KULT modules (user library and user modules). When ordered with a Keithley Model 4200-SCS system, the software is installed at the factory.*

Managing user libraries

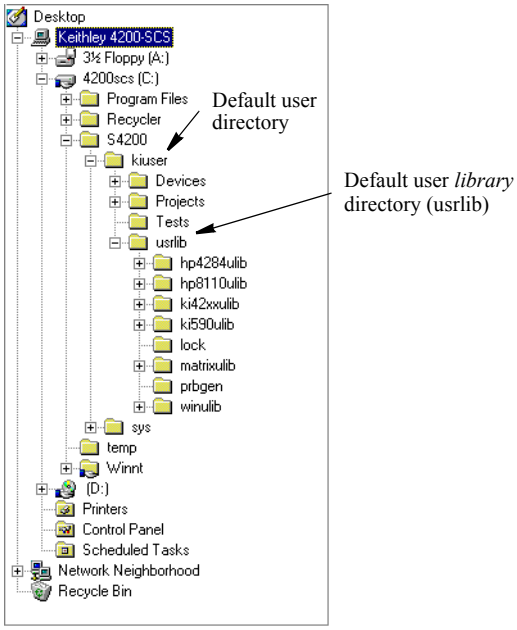
This subsection addresses the following topics:

- Controlling where user libraries are stored describes how to add user libraries at alternative locations: Personal user libraries and network-shared user libraries.
- Changing the active user-library directory describes how to change the directory where KULT and KITE look for user libraries.
- Updating and copying user libraries using KULT command-line utilities describes, in more detail, two command-line utilities. One utility provides a command-line method to copy user libraries. The other utility provides an essential means to update user libraries after they are copied.
- Performing other KULT tasks using command-line commands describes a series of command-line commands. These commands can be used individually or in a batch file to perform various KULT tasks without opening the KULT graphical user interface (GUI).

Controlling where user libraries are stored

When the KTE Interactive software is installed, all user libraries are stored in the `C:\S4200\kiuser\usrlib` directory. Figure 8-41 highlights this directory.

Figure 8-41
Default C:\S4200\kiuser\usrlib active user-library directory



By default, this directory, generically referred to as the KITE / KULT User Library Directory, is the active user-library directory (the directory where KITE and KULT look, exclusively, for user libraries and user modules). However, an alternative directory may be selected instead as the active directory. The following actions apply to the active user-library directory:

- When you edit and compile a user module and build a user library, KULT looks in this directory, exclusively.
- When connecting a user module to a C: directory, KITE allows you to select a user library and user module specifically (and exclusively) from this directory.
- When you execute a UTM, KITE looks in this directory.

The ability to work with user libraries in alternative directories is often desirable, as in the following two situations:

- Multiple users share a Model 4200-SCS. It is desirable that each user works with a personal library, which is stored in a separate location.
- Multiple Model 4200-SCS instruments are installed on a local area network (LAN). It is desirable for all users to be able to access a single user library that is stored on the server of the LAN.

The KTE Interactive software allows you to add libraries at alternative locations, as well as to access these libraries as follows:

- The KTE Interactive software stores the user-library directory path (for example, the default `C:\S4200\kiuser\usrlib)` in an environment variable, `%KI_KULT_PATH%`.[1]
- The KTE Interactive software allows you to change the content of `%KI_KULT_PATH%` to another directory path, using the Keithley CONfiguration (KCON) program that comes with

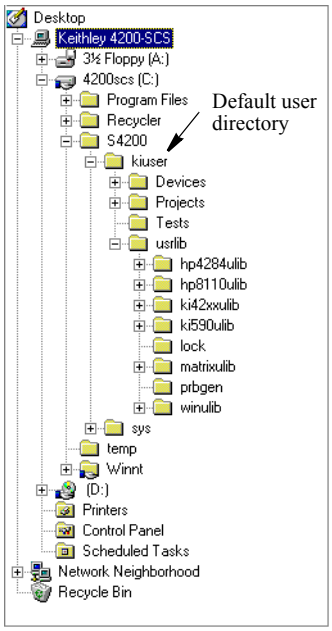
1. `%KI_KULT_PATH%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. For example, the content of `%KI_KULT_PATH%` is the location where KITE and KULT look for user libraries and user modules. The default content of `%KI_KULT_PATH%` is `C:\S4200\kiuser\usrlib`. Use KCON or the `set` command-line utility to change the content of `%KI_KULT_PATH%` to another location, for example, to a personal user-library location, such as C:\S4200\YourName\usrlib. For more information about changing the content %KI_KULT_PATH%, refer to Changing the active user-library directory later in this section.

your Model 4200-SCS. Thereby, you can set an alternative user library to be the active user-library directory.

Adding directories that contain personal user libraries

By default, all user libraries and projects are stored in the `C:\S4200\kiuser` user directory. See Figure 8-42.

Figure 8-42
Default project and user-library directory



However, when the Model 4200-SCS is used in a multi-user environment, all user libraries and projects may also be stored in unique locations for each user. To set up these individual locations most easily, do the following:

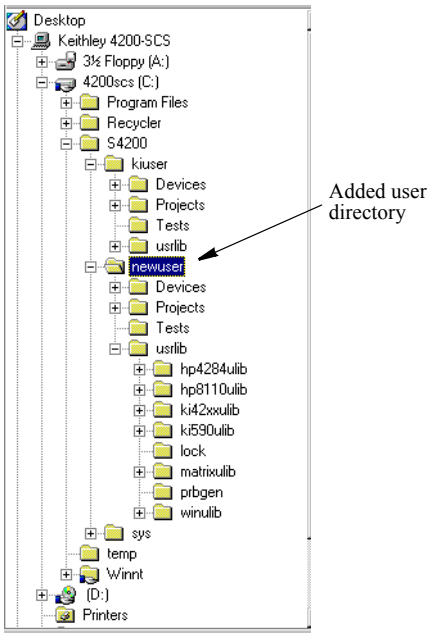
1. Create a new user directory under `C:\S4200`.

NOTE *Suggestion: Give the directory the same name as the user.*

---

2. Copy the contents of the `C:\S4200\kiuser` directory to the Windows clipboard.
3. Paste the contents of the clipboard into the new directory. Figure 8-43 shows a directory created for an individual user called `newuser`.

Figure 8-43
Added personal user directory



4. Before using the user libraries that were copied from `kiuser` to the new directory `newuser`, update them as follows:
   a. Set `%KI_KULT_PATH%` to `C:\S4200\newuser\usrlib` (see Changing the active user-library directory later in this section).
   b. Execute the `kultupdate` command-line utility. Enter the following commands at the command line:

```
C:\>kultupdate Winulib
C:\>kultupdate matrixulib
C:\>kultupdate ki590ulib -dep Winulib
C:\>kultupdate ki42xxulib
C:\>kultupdate hp8110ulib
C:\>kultupdate hp4980ulib
```

NOTE *The* `ki590ulib` *user library depends on the* `Winulib` *user library. Therefore, the* `-dep <library_name>` *option is required in the* `kultupdate` *command for* `ki590ulib`*. None of the other Keithley Instruments-supplied user libraries have dependencies.*

For details about the `kultupdate` *utility, refer to Updating user libraries using kultupdate later in this section.*

5. Repeat steps 1 through 4 for each new Model 4200-SCS user.

CAUTION **After setting up multiple user-library directories, users must ensure that they access and work in their own directories to avoid potential errors caused by executing or, worse, editing someone else's user libraries. Therefore, *each time* before using the Model 4200-SCS, *the user must execute KCON and set the* `%KI_KULT_PATH%` variable to the intended user library directory. Refer to Changing the active user-library directory later in this section.**

Adding a directory that contains network shared user libraries

User libraries can be stored on a local area network so that they can be shared. To accomplish this, do the following:

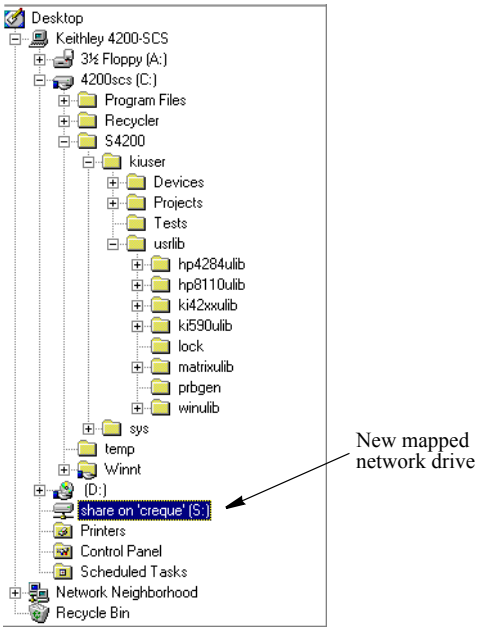
1. Map a network drive with the `net use` command-line utility or through the Windows Explorer (for additional information regarding local area networks, refer to Section 10, System Administration.

   To illustrate, assume that the name of the computer to map is `creque`. Also assume that `creque` has been configured such that its `C:\share` directory is shared. For this scenario, an appropriate `net use` command line would be:

   ```
   C:\net use s: \\creque\share
   ```

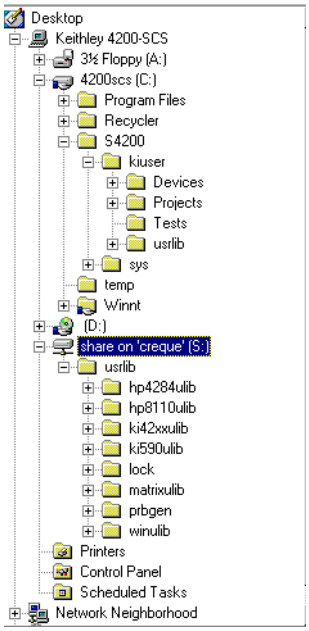
   This command line creates an S: drive that provides access to the `C:\share` directory on the computer named `creque`. See Figure 8-44.

Figure 8-44
New mapped drive, created to provide access to the C:\share directory



2. After mapping the new network drive, copy the `C:\S4200\kiuser\usrlib` folder to the mapped drive. The results of copying `C:\S4200\kiuser\usrlib` to the `S:` drive of the `creque` computer are illustrated in Figure 8-45.

Figure 8-45
New mapped drive, after copying the C:\S4200\kiuser\usrlib folder



3. Before using the user libraries that were copied from `kiuser` to the new mapped drive (drive `S:` in this illustration), update them as follows:
   a. Set `%KI_KULT_PATH%` to `S:\usrlib` (refer to Changing the active user-library directory).
   b. Execute the `kultupdate` command-line utility. Enter the following commands at the command line:

```
C:\>kultupdate Winulib
C:\>kultupdate matrixulib
C:\>kultupdate ki590ulib -dep Winulib
C:\>kultupdate ki42xxulib
C:\>kultupdate hp8110ulib
C:\>kultupdate hp4980ulib
```

NOTE *The* `ki590ulib` *user library depends on the* `Winulib` *user library. Therefore, the* `-dep <library_name>` *option is required in the* `kultupdate` *command for* `ki590ulib`*. None of the other Keithley Instruments-supplied user libraries have dependencies.*

For details about the `kultupdate` *utility, refer to Updating user libraries using kultupdate later in this section.*

All Model 4200-SCS users can now share a common, network accessible user-library directory.

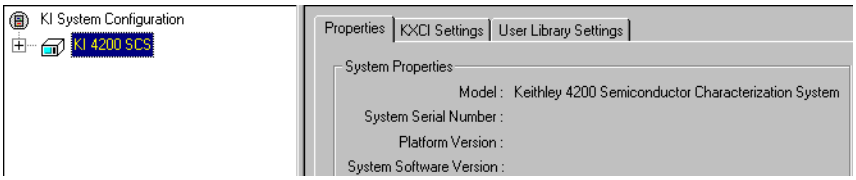
Changing the active user-library directory

The Keithley CONfiguration (KCON) program allows you to change the active user-library directory, which is the directory that is accessed by KULT and KITE.

Change the active user-library directory as follows:

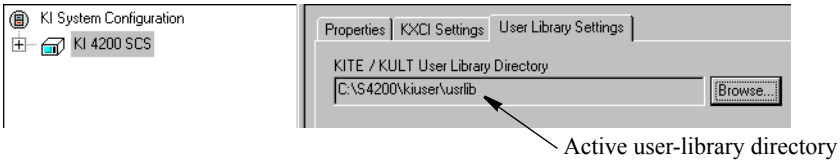
1. Exit KITE and KULT if they are running (select **File > Exit** in each program).
2. Start KCON by double-clicking on the KCON icon or by selecting **Start > Programs > Keithley > KCON.**
3. In the KCON window, single-click on the **KI 4200 SCS** node in the Configuration Navigator. A series of tab selections appears. See Figure 8-46.

Figure 8-46
KCON tab selections



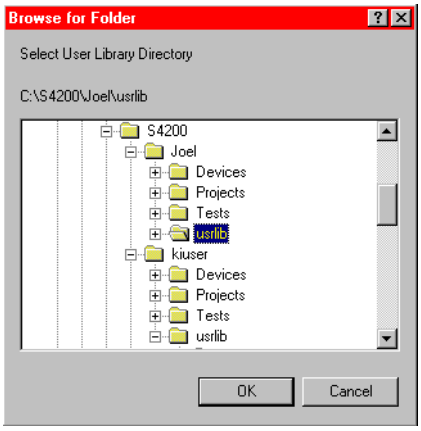
4. Single-click on the **User Library Settings** tab in the work area. The active user-library directory is displayed, as illustrated in Figure 8-47.

Figure 8-47
Active user-library directory in the KCON User Library Settings tab area



5. Single-click on the **Browse** button next to the KITE / KULT User Library Directory field. The **Browse for Folde**r window appears, highlighting the active user-library (**usrlib)** directory. See Figure 8-48.

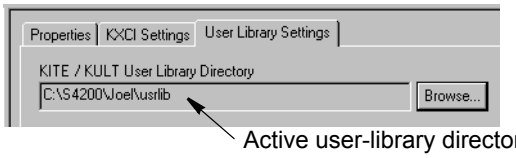
Figure 8-48
Selecting a new user library directory



6. In the **Browse for Folder** window, select the desired user-library directory (**usrlib**). Figure 8-48 shows the **Browse for Folder** window after expanding the **Joel** user directory and highlighting the corresponding **usrlib** directory.

7. Click on **OK**. The new active user-library directory is displayed, as illustrated in Figure 8-49.

Figure 8-49
New active user-library directory in the KCON User Library Settings tab area



8. In the **File** menu, select **Save Configuration.** The path for the new active user library is now stored in `%KI_KULT_PATH%`.[2] KITE and KULT will now access user libraries and user modules exclusively from this directory.
9. In the **File** menu, select **Exit**. KCON closes.

NOTE *Changes to the active user library have no effect on subsequent operations in previously opened command-prompt windows. The user-library changes are effective only in operations that are performed in newly opened command-prompt windows.*

Updating and copying user libraries using KULT command-line utilities

This subsection describes two useful command-line utilities, `kultupdate` and `kultcopy`.

Updating user libraries using kultupdate

The `kultupdate` utility must be used to update user libraries after they have been copied to a new storage location (that is, to another user directory or drive). User libraries must be updated to ensure the correctness of all path information, which is built into the library. The `kultupdate` utility recompiles each user module in the library and rebuilds the library as well.

Usage

```
kultupdate <library_name> [options]
```

Options

Any of the following options may be placed at the `[options]` position in the command:

- `-dep <library_dep_1>...[library_dep_6]`
  Specifies up to six libraries upon which `library_name` depends.
- `-hide`
  Hides `library_name` so that it is not visible in KITE.
- `+hide`
  Unhides `library_name` so that it is visible in KITE.

Example

Update the `ki590ulib` library (in the active user-library directory), which depends on the `Winulib` library.

2. `%KI_KULT_PATH%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. For example, the content of `%KI_KULT_PATH%` is the location where KITE and KULT look for user libraries and user modules. The default content of `%KI_KULT_PATH%` is `C:\S4200\kiuser\usrlib`. Use KCON or the set command-line utility to change the content of `%KI_KULT_PATH%` to another location, for example, to a personal user-library location such as `C:\S4200\YourName\usrlib`.

```
C:\>kultupdate ki590ulib -dep Winulib
```

Copying user libraries using kultcopy

The `kultcopy` utility copies any user library from any accessible storage location to the active user-library directory. More specifically, `kultcopy` does the following:

- Copies the user library that is specified by the "Start-In" user-library directory, which is the directory in which you start the `kultcopy` command. Refer to the NOTE below, to the example below, and to Changing the active user-library directory earlier in this section.

NOTE *To successfully copy a user library to the active user-library directory, you must start* `kultcopy` *in the following directory:*

<source_lib_path>\<source_lib_name>\src

This directory is called the "Start-In" directory, where:

- <source_lib_path> *is any accessible user library directory.*
- <source-lib-name> *is the name of the specific user library to be copied.*

For instructions on changing the destination directory (`%KI_KULT_PATH%`*), refer to Changing the active user-library directory earlier in this section.*

- Performs `kultupdate` so that the user library is immediately ready for use (refer to Updating user libraries using kultupdate earlier in this section).

Usage

```
kultcopy <library_name> [options]
```

Options

Any of the following options may be placed at the `[options]` position in the command:

- `-dep <library_dep_1>...[library_dep_6]`
  Specifies up to six libraries upon which `library_name` depends.
- `-hide`
  Hides `library_name` so that it is not visible in KITE.
- `+hide`
  Unhides `library_name` so that it is visible in KITE.

Example

If both of the following are true:

- The active user-library directory is `C:\S4200\newuser\usrlib` (the `%KI_KULT_PATH%` variable is equal to `C:\S4200\newuser\usrlib`)
- The `kultcopy` utility is started in the `C:\S4200\kiuser\usrlib\Winulib\src` directory

then the following command:

```
C:\S4200\kiuser\usrlib\Winulib\src>kultcopy Winulib
```

"Start-In" directory

copies the Winulib user library from the location C:\S4200\kiuser\usrlib to the new location, C:\S4200\newuser\usrlib, and automatically updates it.

Performing other KULT tasks using command-line commands

The KULT command-line interface lets you load, build, or delete user libraries and add or delete user modules without opening the KULT graphical user interface (GUI). This feature is useful when developing and managing user libraries. The commands can be used individually or in a batch file.

The general format for a command line instruction is as follows:

```
kult subcommand -l<library_name> [options] [module]
```

The individual items in the instruction are as follows:

- The item subcommand may be any one of these subcommands:
  - new_lib
  - del_lib
  - add_mod
  - compile_mod
  - bld_lib
- The item <library_name> specifies the name of the library involved in the commanded action.
- The item [options] includes one or more of these options:
  - -d<directory_name>
  - -hide
  - +hide
  - -dep <library_dep_1>.....[library_dep_6]

  These options are described in individual subcommand subsections below.
- If appropriate to the commanded action, [module] specifies the name of the involved user module.

The subsections that follow describe the five subcommands.

The new_lib subcommand

The new_lib subcommand lets you create a new user library without any user modules. Its action is equivalent to the following steps in KULT:

- Starting KULT
- Selecting **File > New Library**
- Entering a new library name
- Clicking **OK**
- Selecting **File > Exit**

Usage

```
kult new_lib -l<library_name>
```

The <library_name> user library is created in the active user-library directory.

The del_lib subcommand

The `del_lib` subcommand lets you delete a library from the command line. Its action is equivalent to the following steps in KULT:

- Starting KULT
- Selecting **File > Delete Library**
- Selecting a user library to be deleted
- Clicking **OK**
- Selecting **File > Exit**

Usage

```
kult del_lib -l<library_name>
```

The `<library_name>` user library is deleted from the active user-library directory.

The add_mod subcommand

The `add_mod` subcommand lets you add (or copy) a user module from one user library (source) to another library (target). Its action is equivalent to the following KULT steps:

- Starting KULT
- Selecting **File > Open Library**
- Selecting the `<source_lib_name>` source library
- Selecting **File > Open Module**
- Selecting the `<module>` source module
- Selecting **File > Copy Module**
- Selecting the `<library_name>` target library
- Entering a target-module name

NOTE *All user modules must be named uniquely, even if they are duplicates that reside in different user libraries. The* `add_mod` *subcommand automatically assigns a target-module name that is a derivative of the source-module name. The naming convention is as follows:* `<source_library_name>_<module>`.

- Selecting **File > Exit**

Usage

```
kult add_mod -l<library_name> [-d<source_lib_path>\source_lib_name>\src] <mod-
ule>
```

Where:

- `<library name>` is the target library into which `<module>` is to be copied. It must be located in the active user-library directory.
- `<source_lib_path>` is any accessible user-library directory.
- `<source_lib_name>` is the name of the specific user library from which <module> is to be copied.
- `<module>` is the source user module.

You must use the -d option when you execute add_mod in a directory other than `<source_lib_path>\<source_lib_name>`.

The compile_mod subcommand

The `compile_mod` subcommand lets you compile a user module in an existing user library. Its action is equivalent to the following KULT steps:

- Starting KULT
- Selecting **File > Open Library**
- Selecting `<library_name>`, the library that contains the module to be compiled
- Selecting **File > Open Module**
- Selecting `<module>`, the name of the module to be compiled
- Selecting **Options > Compile**
- After the module compiles, selecting **File > Exit**

Usage

```
kult compile_mod -l<library_name> <module>
```

Compiles the `<module>` module in the `<library_name>` user library, which is located in the active user-library directory.

The bld_lib subcommand

The `bld_lib` subcommand lets you build a user library from the command line. Its action is equivalent to the following steps in KULT:

- Starting KULT
- Selecting **File > Open Library**
- Selecting the `<library_name>` user library
- Clicking **OK**, selecting **Options > Build Library**
- After the build is completed, selecting **File > Exit**

Usage

```
kult bld_lib -l<library_name> [options]
```

Builds the `<library_name>` user library in the active user-library directory.

Options

Any of the following may be placed at the `[options]` position in the command:

- `-dep <library_dep_1>...[library_dep_6]`
  Specifies up to six user libraries upon which `library_name` depends.

NOTE *Dependent user libraries must be located in the active user-library directory. For more information about dependent libraries, refer to Working with interdependent user modules and user libraries below.*

- `+hide`
  Hides `library_name` so that it is not visible in KITE.
- `-hide`
  Unhides `library_name` so that it is visible in KITE.

Working with interdependent user modules and user libraries

KULT allows a user module to call other user modules. A called user module may be located within the same user library as the calling module, or may be located in another user library. When the module that you are creating calls a module in another user library, you must 1) select **Library**

Dependencies in the **Options** menu and 2) specify each called library from the list that is displayed.

You must select user module and user-library dependencies carefully. Observe the following:

- Try to put user modules with interdependencies in the same user library and minimize the number of interdependencies between libraries. This practice helps to avoid problematic user library dependency loops (`Lib1` relies on `Lib2`, `Lib2` relies on `Lib3`, `Lib3` relies on `Lib1`).
- If a user module in one user library must depend on user modules in other user libraries, take care when selecting the user libraries to be linked with the user module under development. The next subsection provides guidance.

NOTE *The user libraries to be linked are saved such that future rebuilds do not require the dependencies to be reselected. This information is stored in the* `<library_name>_modules.mak` *file located in the* `%KI_KULT_PATH%\<library_name>\kitt_obj` *directory.*[3]

- Structure dependencies hierarchically to avoid circular dependencies, and then build the dependent user libraries in the correct order. The next two subsections provide the needed guidance.

Structuring dependencies hierarchically

User library circular dependency can be avoided by calling user libraries in a hierarchical design, as illustrated in Figure 8-50.

Observe the following:

- Design lower-level user modules in the calling hierarchy so that they do not require support from higher-level modules. That is, lower-level user modules should not require calls to higher-level modules to perform their required tasks.
- Use several general-purpose low-level-library user modules to perform a task rather than a single, do-all, higher-level-library user module.

You may find it helpful to prefix user modules with the user-library name as an identifier, for example, `liba_ModuleName` for user modules contained in `liba`. This avoids duplicate user module names and prevents confusion with similarly named modules contained in other user libraries and source files (when you execute the **File > Library Copy** command, KULT automatically appends the user library name to each user module in the new user library name. KULT also appends the library name, as a suggestion, when you execute the **File > Module Copy** command).

In Table 8-6, the series of coded user modules amplifies the hierarchical dependencies shown in Figure 8-50.

3. `%KI_KULT_PATH%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. For example, the content of `%KI_KULT_PATH%` is the location where KITE and KULT look for user libraries and user modules. The default content of `%KI_KULT_PATH%` is `C:\S4200\kiuser\usrlib`. Use KCON or the `set` command-line utility to change the content of `%KI_KULT_PATH%` to another location: for example, to a personal user-library location, such as `C:\S4200\YourName\usrlib`. For more information about changing the content `%KI_KULT_PATH%`, refer to Changing the active user-library directory earlier in this section.

Table 8-6
Coded user modules illustrating the use of hierarchical user library dependencies

Hierarchy level	User-library name	User-module name	User-module code

0	liba	Test	void Test(void)
			{
			  printf("In liba, calling CalledA1()\n");
			  CalledA1();
			}

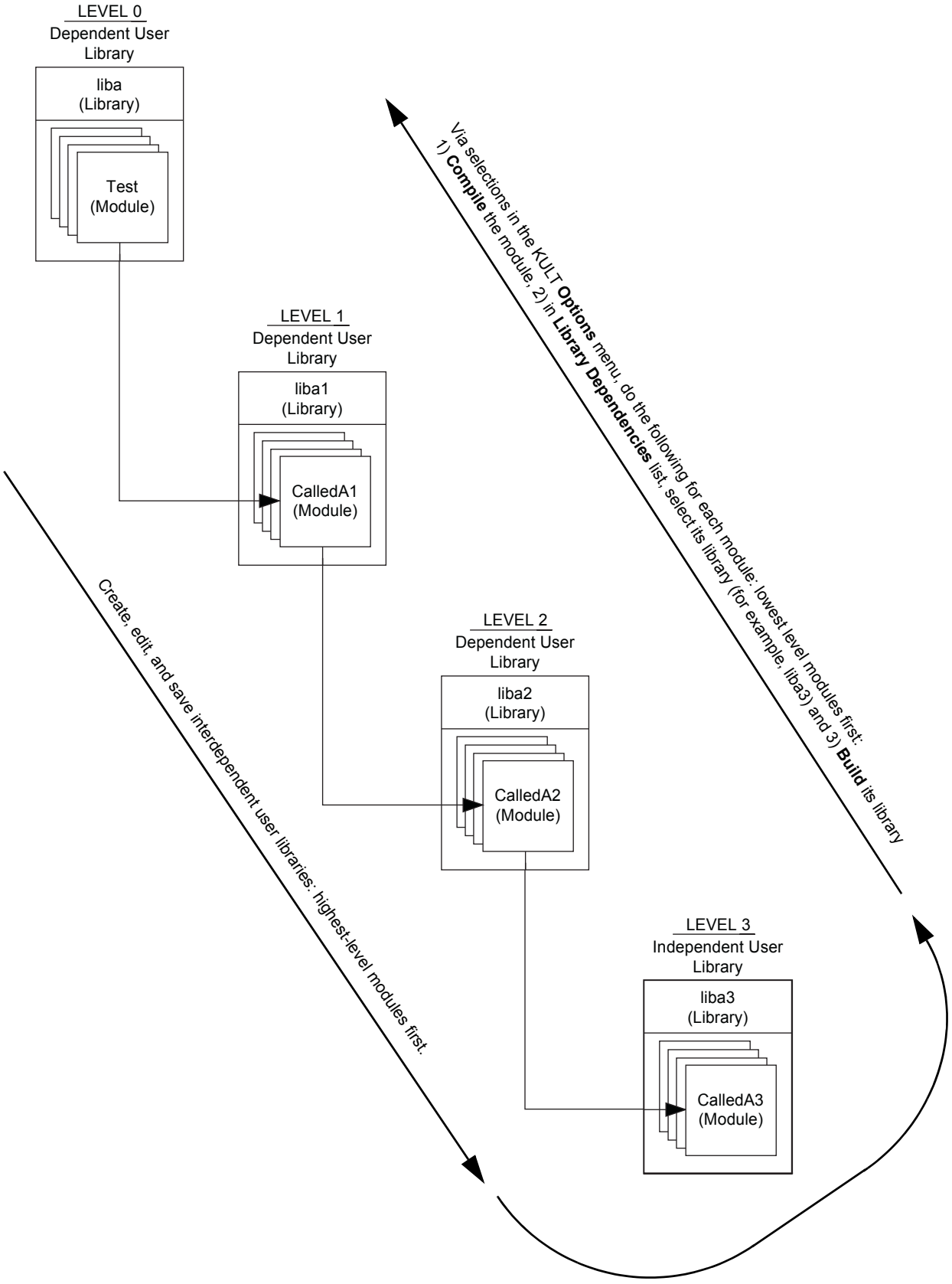
1	liba1	CalledA1	void CalledA1(void)
			{
			  printf("In liba1, calling CalledA2()\n");
			  CalledA2();
			}

2	liba2	CalledA2	void CalledA2(void)
			{
			  printf("In liba2, calling CalledA3()\n");
			  CalledA3();
			}

3	liba3	CalledA3	void CalledA3(void)
			{
			  printf("In liba3, making no calls()\n");
			}

A user module in `liba` calls a user module located in `liba1`. In turn, a user module in `liba1` calls a user module located in `liba2`. Finally, a user module in `liba2` calls a user module located in `liba3`.

Figure 8-50
Hierarchical design for user library dependencies



Building dependent user libraries in the correct order

When KULT builds a user library that depends on other user libraries, it must link to each of these libraries. For example, when KULT builds `liba`, the following linkages occur: `liba` is linked with `liba1`, the `liba/liba1` pair is linked with `liba2`, the `liba/liba1/liba2` trio is linked with `liba3`, and so on. Therefore, a series of hierarchical dependencies requires a reverse hierarchical build order, starting first with the lowest-level user library. That is, before building any dependent user library, you must first successfully build each library on which it depends, as illustrated below:

- If `liba` depends on `liba1`, `liba` cannot successfully build until `liba1` has been built.
- If, additionally, `liba1` depends on `liba2`, both `liba` and `liba1` cannot successfully build until `liba2` has been built.
- Finally, if `liba2` depends on `liba3`, then the three higher level user libraries (`liba`, `liba1`, and `liba2`) cannot successfully build until `liba3` has been built.

The following procedure[4] illustrates the correct reverse build order for the dependencies shown in Figure 8-50 and Table 8-6:

1. Compile and build the Level 3 user module and user library.
   a. Compile the saved `CalledA3` user module, located in the `liba3` user library (select **Compile** in the KULT **Options** menu).
   b. Build the `liba3` user library (select **Build** in the KULT **Options** menu).
2. Compile, set dependencies for, and build the Level 2 user module and user library:
   a. Compile the saved `CalledA2` user module, located in the `liba2` user library.
   b. Set the dependencies for the `CalledA2` user module.
      1) Select **Library Dependencies** in the **Options** menu.
      2) Select **liba3** from the Library Dependencies list box (displayed by selecting Library Dependencies in the **Options** menu).
      3) Click **Apply**.
   c. Build the `liba2` user library.
3. Compile, set dependencies for, and build the Level 1 user module and user library:
   a. Compile the saved `CalledA1` user module, located in the `liba1` user library.
   b. Set the dependencies for the `CalledA1` user module:
      1) Select **Library Dependencies** in the **Options** menu.
      2) Select **liba2** from the Library Dependencies list box.
      3) Click **Apply**.
   c. Build the `liba1` user library.
4. Compile, set dependencies for, and build the Level 0 user module and user library:
   a. Compile the saved `Test` user module, located in the `liba` user library.
   b. Set the dependencies for the `Test` user module:
      1) Select **Library Dependencies** in the **Options** menu.
      2) Select **liba1** from the Library Dependencies list box.
      3) Click **Apply**.
   c. Build the `liba` user library.

This reverse hierarchical build order results in a linking scheme that satisfies the dynamic linking requirements of Windows®.

4. This is a general procedure based on the assumption that each of the interdependent user modules are newly created or were edited since the last compile and build. Compiles and builds that are already complete up to a given level of dependency need not be repeated.

Understanding user module locking

Edit locking

When user libraries are stored on a network, they can be shared between multiple users who are operating multiple Model 4200-SCS systems. However, in such a situation, multiple users can simultaneously edit the same user module and then attempt to save it under the same name. To avoid such conflicts, when a user module is first opened, KULT creates a temporary lock file in the directory `%KI_KULT_PATH%\<library name>\lock`.[5] This lock file prevents other users from saving the user module while it is open, unless they rename it (another user may still access and edit the user module but cannot save the edited module without changing its name).

Effects of edit lock files

When a user tries to access a user module that is already open, a message box appears stating that another user has locked the module. For example, if two users attempt to access the `VSweep` user module in the `my_2nd_lib` user library (created in User's Manual, Tutorial 2: Creating a user module that returns data arrays, page 6-28), the second user sees a message similar to the one in Figure 8-51.

Figure 8-51
Example of edit-lock caution message

After the second user clicks **OK**, the user module opens normally. However, if the second user edits the module while it is locked by the first user, the second user must save it under a new name, using the KULT **File > Copy Module** menu selection. The new name must be unique; it cannot be the name of any other user module that is located in the `%KI_KULT_PATH%` directory. Otherwise, KULT will not allow the user module to be saved.

Edit lock-file naming and content

The lock file is named as follows:

```
%KI_KULT_PATH%\<library name>\lock\<module name>.lck
```

For example, when a user opens the `VSweep` user module in the `my_2nd_lib` user library (created in the User's Manual, Tutorial 2: Creating a user module that returns data arrays, page 6-28), the following lock file is created:

```
%KI_KULT_PATH%\my_2nd_lib\lock\VSweep.lck
```

5. `%KI_KULT_PATH%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. For example, the content of `%KI_KULT_PATH%` is the location where KITE and KULT look for user libraries and user modules. The default content of `%KI_KULT_PATH%` is `C:\S4200\kiuser\usrlib`. Use KCON or the `set` command-line utility to change the content of `%KI_KULT_PATH%` to another location, for example, to a personal user-library location, such as `C:\S4200\YourName\usrlib`. For more information about changing the content `%KI_KULT_PATH%`, refer to Changing the active user-library directory earlier in this section.

An example of this lock file, which stores information textually, contains the following:

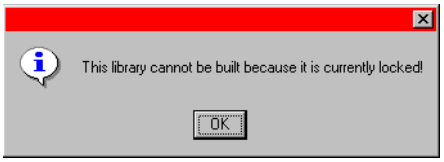
```
PID:162
USER:kiadmin
HOST:S4200-P3
TIME:Friday, 03/17/2000, 17:28:40
LIB:my_2nd_lib
MOD:VSweep.c
FILE:C:\S4200\kiuser\usrlib\my_2nd_lib\lock\my_2nd_lib_VSweep.lck
```

NOTE *KULT automatically deletes an edit lock file when the corresponding user module is closed.*

Run-time locking

It is possible to edit, save, and compile any user module in the user library while one of its user modules is being run. However, run-time user-library locking prevents building of a KULT user library while one of its user modules is being run in KITE. If you try, you see the message shown in Figure 8-52.

Figure 8-52
Run-time lock message



Run-time lock files are generated in the following directory:

```
%KI_KULT_PATH%\lock
```

When KITE executes a UTM, a run-time lock file is automatically created during the normal loading of the user module. The run-time lock file is automatically deleted at the end of the run during normal unloading of the user module.

Removing locks that remain after interrupted operation

If normal operation of KULT or normal execution of user libraries is interrupted, the protective locks may not be removed automatically. However, you can remove locks manually via the `kultcleanlocks` command-line utility. This utility deletes all lock files located in both of the following directories:

- The `%KI_KULT_PATH%\lock` directory
- The `%KI_KULT_PATH%\<library name>\lock` directories

Execute the `kultcleanlocks` utility as follows:

1. Click on the **Command Prompt** icon on the desktop or in the **Start > Programs** menu. The **Command Prompt** window opens.
2. At the **Command Prompt**, type `kultcleanlocks`.
3. Press **ENTER**. The `kultcleanlocks` utility executes and deletes all residual lock files.

Debugging user modules using Microsoft Visual C++

NOTE *Microsoft Visual Studio is available from Keithley Instruments as an accessory.*

At times you may wish to perform step-by-step debugging of a library user module, using the capabilities of Microsoft Visual C++. To facilitate this process, Keithley Instruments provides the `create_dt` command-line utility. The `create_dt` utility automatically generates a small Visual C++ program, called a debug task, in which to test/debug your module.

Creating a debug task

To create a Visual C++ debug task using the `create_dt` utility, do the following:

1. At a command prompt, enter `create_dt` followed by the name of the debug task, as follows:

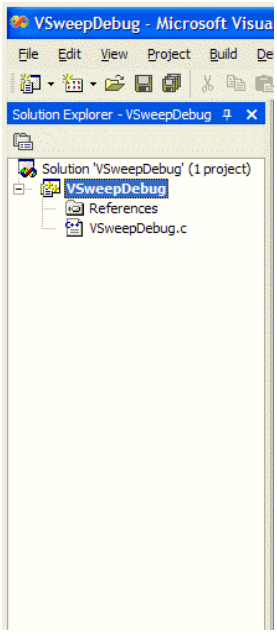
   ```
   C:\> create_dt <debugtaskname>
   ```

   For example, the `VSweep` user module that was created in the User's Manual, Tutorial 2: Creating a user module that returns data arrays, page 6-28 could be debugged in a debug task called `VSweepDebug`. To create the debug task, you would enter the following at the command line prompt:

   ```
   C:\> create_dt VSweepDebug
   ```

2. Press **ENTER**.
   - The `create_dt` utility first generates a debug-task main program and a Microsoft Visual C++ project file. These files are placed in the `%KIPGM%`[6] subdirectory.
   - Next, the utility starts the Microsoft Visual C++ development environment.
3. Figure 8-53 shows the Solution Explorer area for the `VSweepDebug` debug task.

Figure 8-53
Visual C++ Solution Explorer area displaying debug-task name



6. `%KIPGM%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. The default content of `%KIPGM%` is `C:\S4200\kiuser\dbtask\`. Use the `set` command-line utility to change the content of `%KIPGM%`.

4. Expand the `<debugtaskname>` files item to display the files in the `<debugtaskname>` Visual C++ project.
5. Double-click on `<debugtaskname>.c`. The Microsoft Visual C++ development environment displays the debug task source code that was generated by the `create_dt` utility. Figure 8-54 shows the debug task code that was created for the `VSweep` user module.
6. Compile the `<debugtaskname>.c` program by pressing the **CTRL** + **F7** keys or by selecting `Compile <debugtaskname>.c` in the **Build** menu.

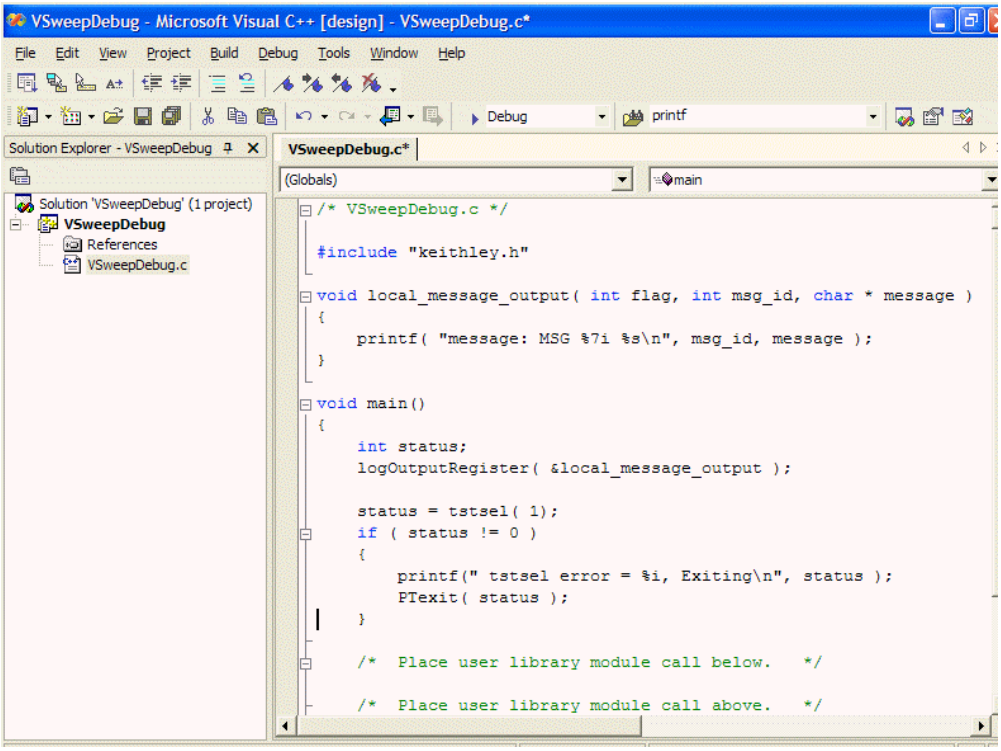
Loading a debug task

To load a saved debug task, use the `load_dt` command-line utility.

```
c:\>load_dt <debugtaskname>
```

For example, to load the debug task called `VSweepDebug`, enter the following at the command line prompt: `c:\>load_dt VSweepDebug`

Figure 8-54
Example code generated by the create_dt utility



UTM GUI view

After you create a user module, you can use it as a user test module (UTM) in a project. Figure 8-55 shows the UTM Definition Tab Classic View; Figure 8-56 shows the UTM GUI Definition View. This topic describes the creation of a graphical user interface (GUI) definition (a viewer) for a UTM (refer to Figure 8-56).

Figure 8-55
Definition tab, Classic View

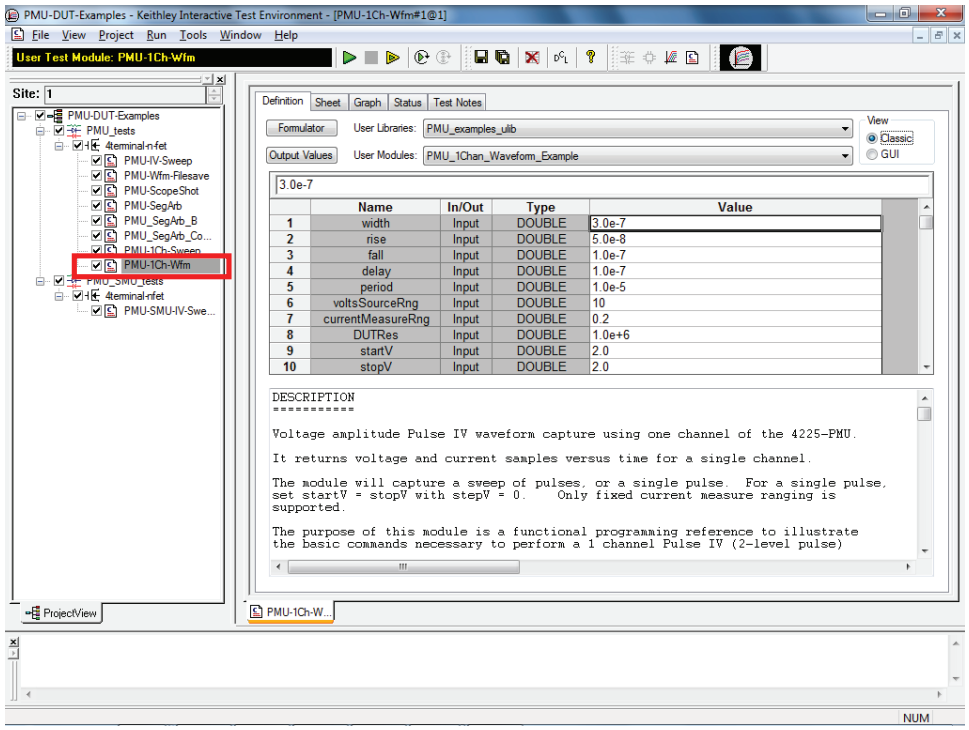
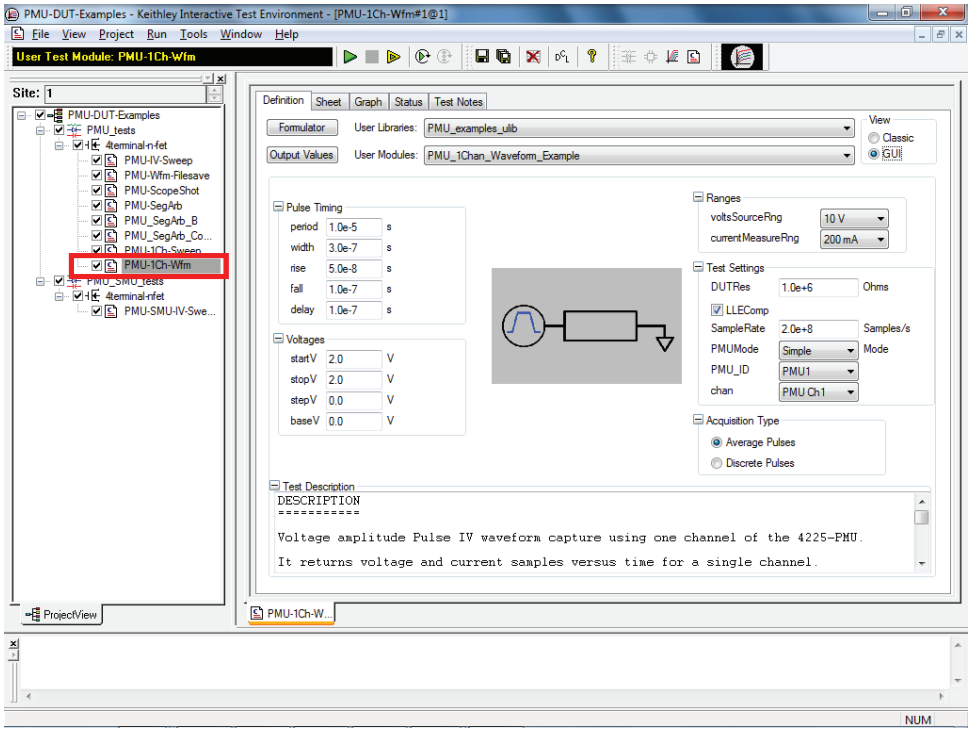


Figure 8-56
Definition tab, GUI View



Generally, creating a UTM GUI view for a UTM makes it easier for a user to configure and use a user module rather than modifying it using the classic UTM definition tab. Therefore, think carefully about how the test will be used and the parameters that should appear on the GUI view.

NOTE *If you change a UTM parameter name (using KULT) after the GUI Definition exists, make sure to update the GUI Definition. This makes sure the new parameter name has a group and is displayed (if you do not update it, the new parameter name will not have a group and not be displayed).*

Creating a UTM GUI definition using the UTM GUI editor

NOTE *A UTM GUI is specific to a user module. Editing a UTM GUI changes the UTM GUI for that user module. To use a different UTM GUI, create a different user module.*

The UTM GUI Definition view graphically represents the user module's parameters. Creating a user view allows you to group the parameters to promote understanding and enhance ease of use of the module. You can customize the image of the test (DUT) to illustrate the overall test capability of the module. To reduce the learning curve of your test further, an optional tooltip text field is available for each parameter. It is important to note that this presentation does not change the test or overall project execution. When creating a user GUI definition, display only the most important or the most commonly used parameters. To configure and optimize a GUI view properly, you need to be familiar with the test details of the user module, its parameters, and which parameters are commonly changed. For a basic illustration and explanation of the UTM Definition tab fields, you can refer to “Defining a UTM” on page 6-19.

NOTE *The UTM GUI views created using the UTM GUI Editor are part of KITE (not KULT).*

To find out the relationship between the classic and GUI UTM views, look at the "Pulse Timing" group in the GUI view (Figure 8-56). This group contains the parameter identity cells for period, width, rise, fall, and delay. Also included in the group are the associated parameter entry cells and units of measure. This compares to the Classic view in Figure 8-55. To find the parameter identity cells associated with pulse timing in the classic view, you must know the specific parameter identity name. To modify the parameter, you must also know the specific parameter's unit of measure.

NOTE *A user module must already exist before a GUI view can be created or modified with the UTM GUI Editor. Create (or modify) user modules by using KULT (Keithley User Library Tool) as previously described.*

There are three ways for the GUI view to display the test parameters of an existing user module:

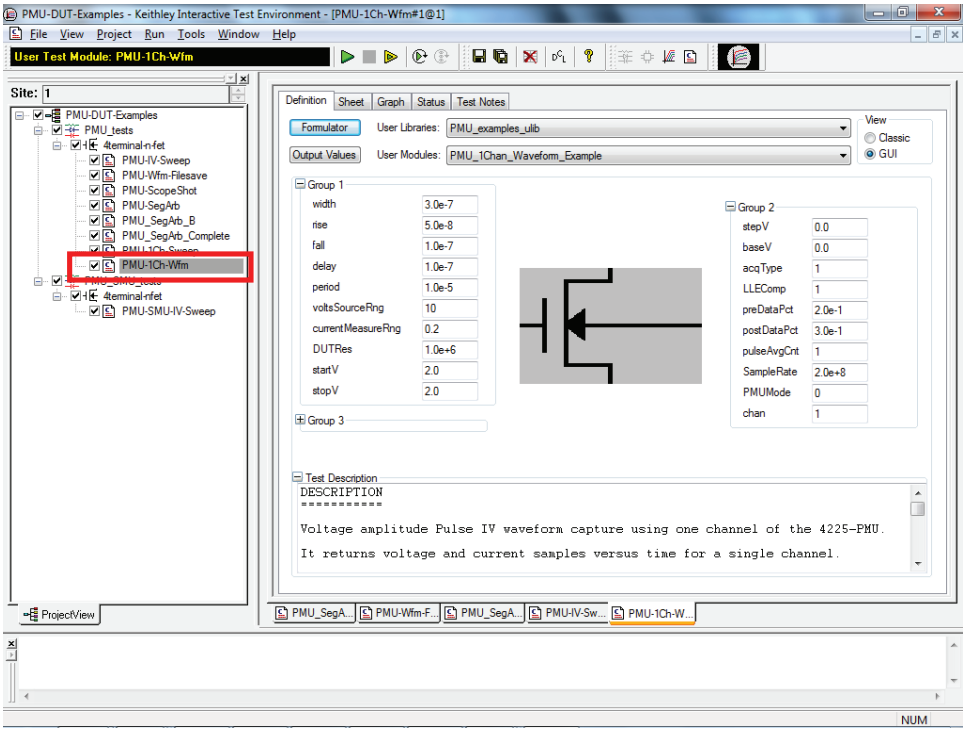
Factory: Many user modules supplied with the Model 4200-SCS include factory UTM GUI definitions. For an example, see Figure 8-56.

User: You can create GUI definitions for user modules. You can create user definitions to replace the factory-supplied GUI views, or for user modules where no GUI definition exists.

Default: If no GUI definition exists, a GUI presentation is dynamically created. The parameters are placed in groups around a default image of the device under test (DUT) graphic. Figure 8-57 illustrates the same user module as Figure 8-56 but with a dynamically generated version of the

UTM GUI view. The groups in the dynamically created view are named "Group 1" and "Group 2." Each group contains up to ten parameters. Figure 8-57 displays the default image of a 4-terminal-n-fet.

Figure 8-57
Dynamically generated UTM GUI



If a user GUI definition for the module exists, it is used. If no user definition exists, KITE attempts to find a factory definition. If no factory definition (or user definition) is found, KITE dynamically creates a GUI definition.

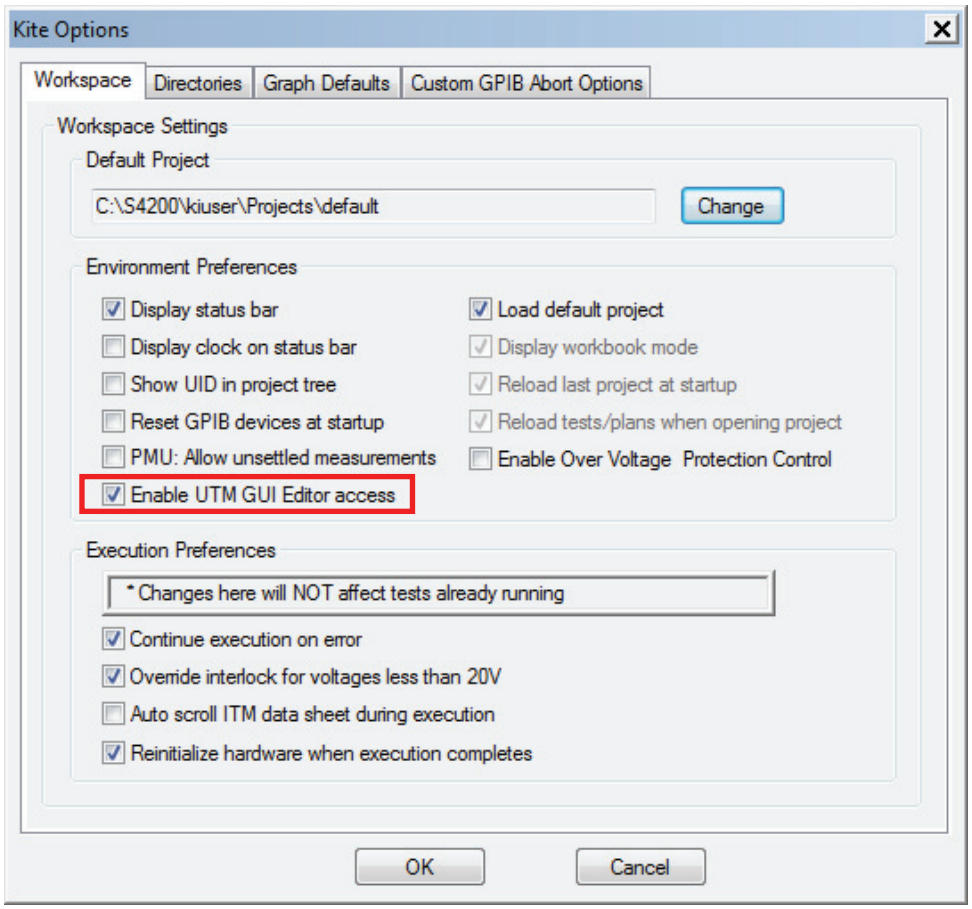
Enabling access to the UTM GUI Editor

NOTE *The UTM GUI Editor is part of KITE (not KULT).*

To enable the editor in KITE:

1. From the Tools menu, select Options.
2. In the Kite Options window, select Enable UTM GUI Editor access (located in the Environmental Preferences group, Figure 8-58). You can clear this selection later to prevent accidental modification of the UTM GUI definitions.

Figure 8-58
Enable UTM GUI Editor access



Using the UTM GUI Editor

The main GUI Editor dialog (shown in Figure 8-59) allows for configuration of the bitmap image used, as well as displaying a summary of the test parameters and their attributes. This includes groups, control type, minimum, maximum and default values, as well as units of measure and tooltip text. The entire table is a summary and is not editable (see "Editing the attributes for a test parameter" on page 8-69).

The `PMU_1Chan_Waveform_Example` user module from the user library `PMU_examples_ulib` is used to illustrate the GUI editor's operation. In this project, `PMU-DUT-Examples` (located in the `_Pulse` projects directory), look for the UTM `PMU-1Ch-Wfm` as shown in Figure 8-56. Note that this factory UTM GUI view definition comes with KTEI. This view illustrates some of the various controls (edit boxes, list boxes, option buttons, and a check box, see Figure 8-60) that are available when creating a user UTM GUI.

Figure 8-59
Main UTM GUI editor

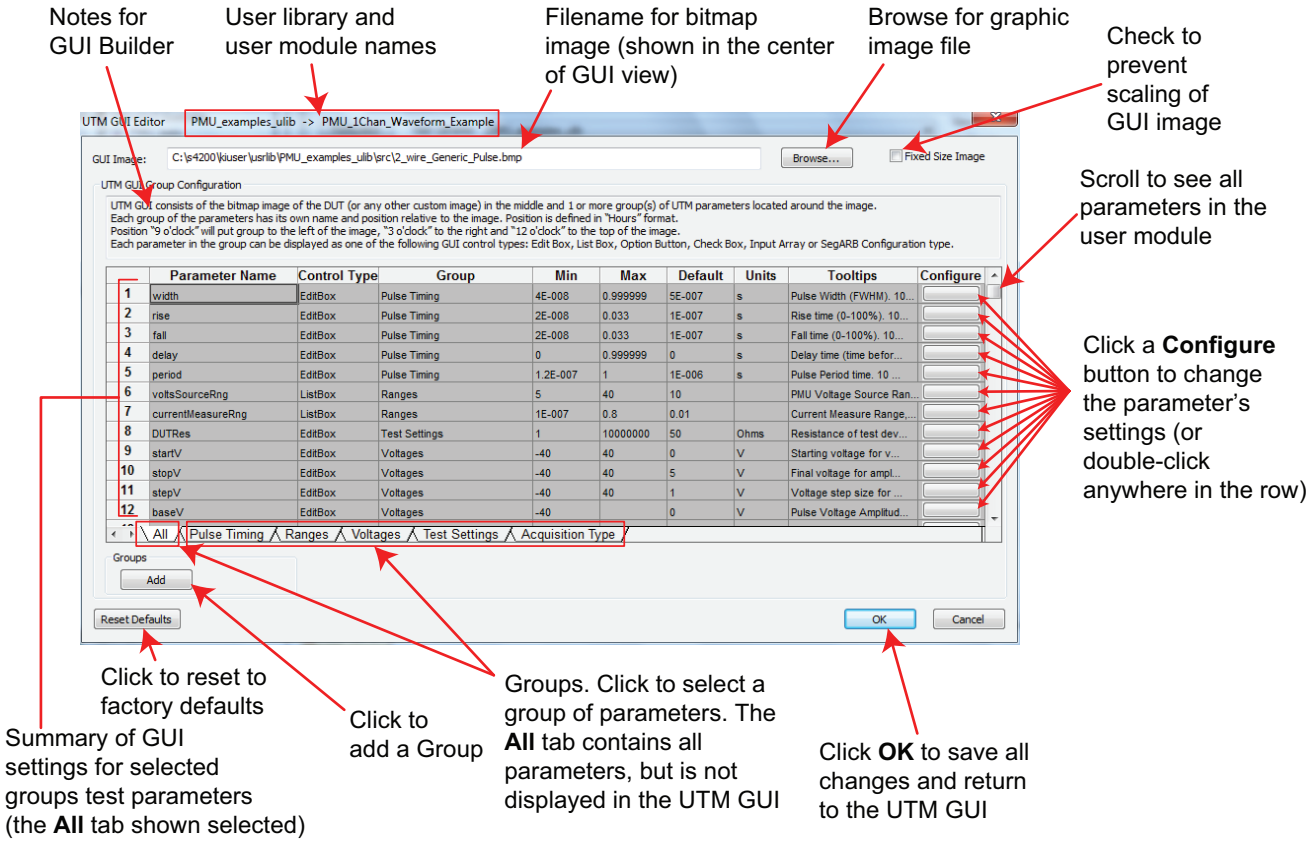
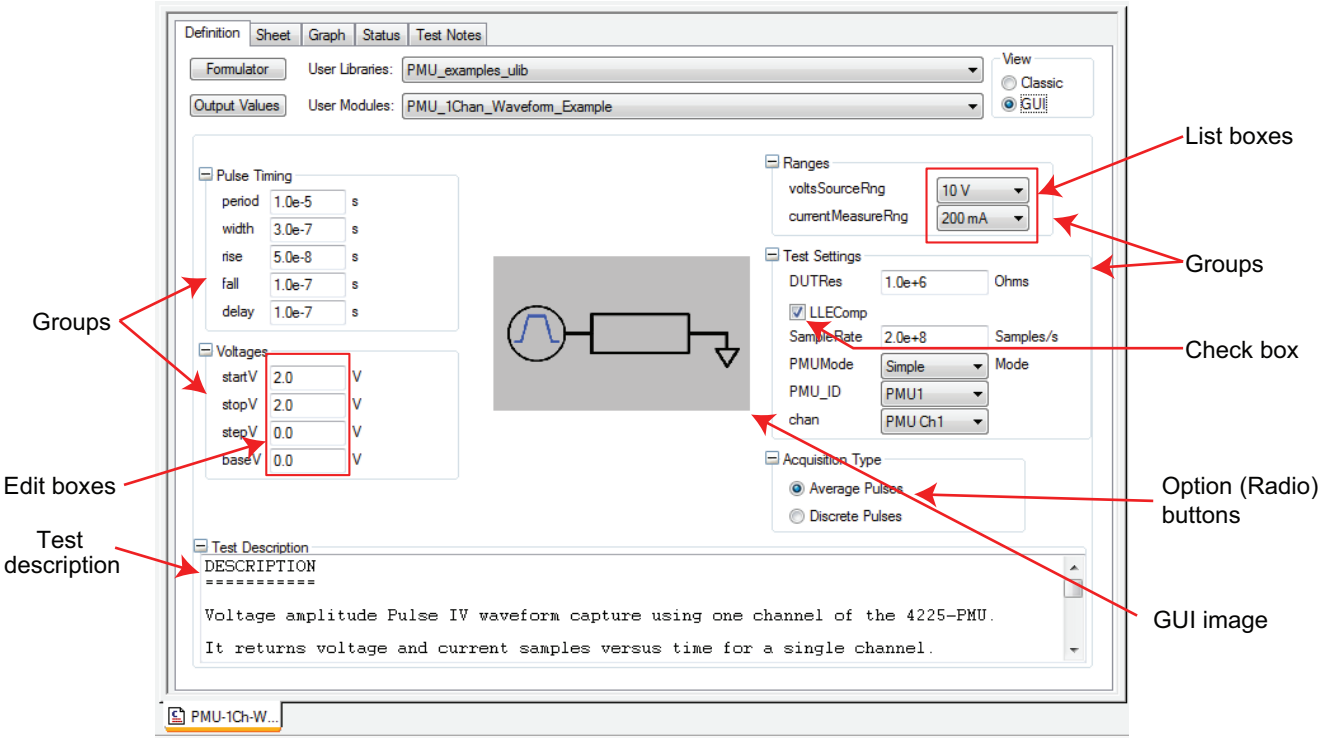


Figure 8-60
GUI view element sample



Example of using the editor

1. Make sure access to the editor is enabled (see "Enabling access to the UTM GUI Editor" on page 8-62).
2. Double-click to select the `PMU-1Ch-Wfm` UTM in the `PMU-DUT-Examples` project, and then right-click anywhere on the definition tab to bring up the Edit UTM GUI menu Figure 8-61).
3. Click the Edit UTM GUI menu item.
4. The Main UTM GUI Editor window will open (Figure 8-59).

Figure 8-61
Starting the UTM GUI editor (right-click on the UTM GUI view)

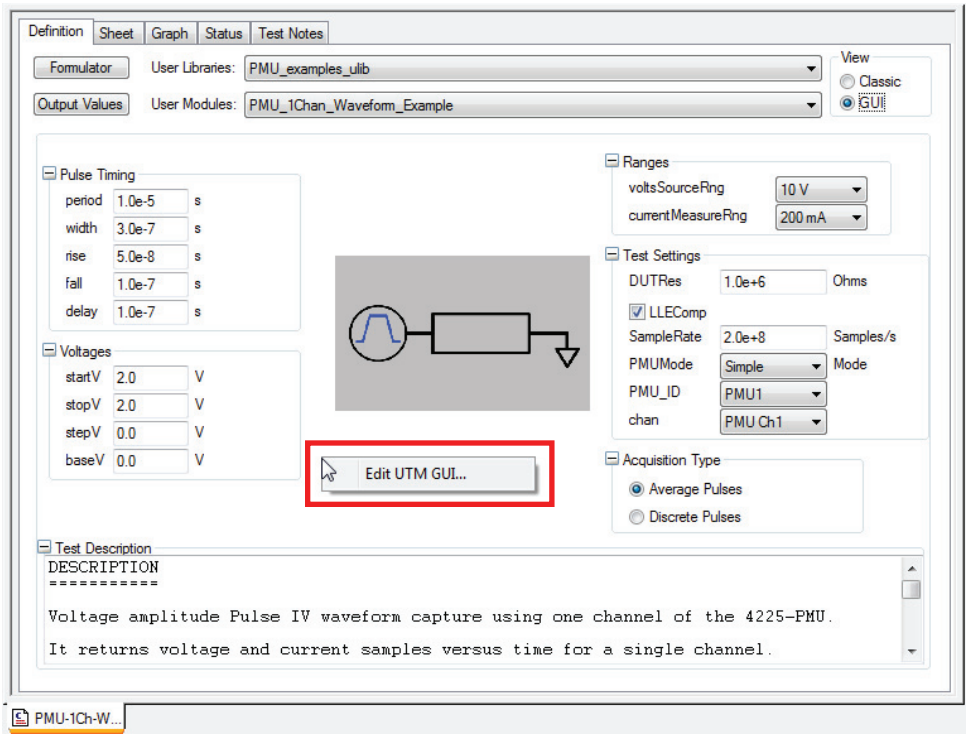


Figure 8-60 illustrates some of the available controls for the UTM GUI view including groups, edit boxes, list boxes, and check boxes. These controls are configured within the UTM GUI Editor (the editor is shown in Figure 8-59).

NOTE *The user module defines the Test Description content (as defined in the KULT description tab); the UTM GUI Editor does not define the Test Description content (for more information, see the "Description tab area" on page 8-10).*

Completing a change

NOTE *A UTM GUI is specific to a user module. Editing a UTM GUI changes the UTM GUI for that user module. To use a different UTM GUI, create a different user module.*

Click the **OK** button to save any changes; click the **Cancel** button to exit the dialog without saving any of the changes.

Groups

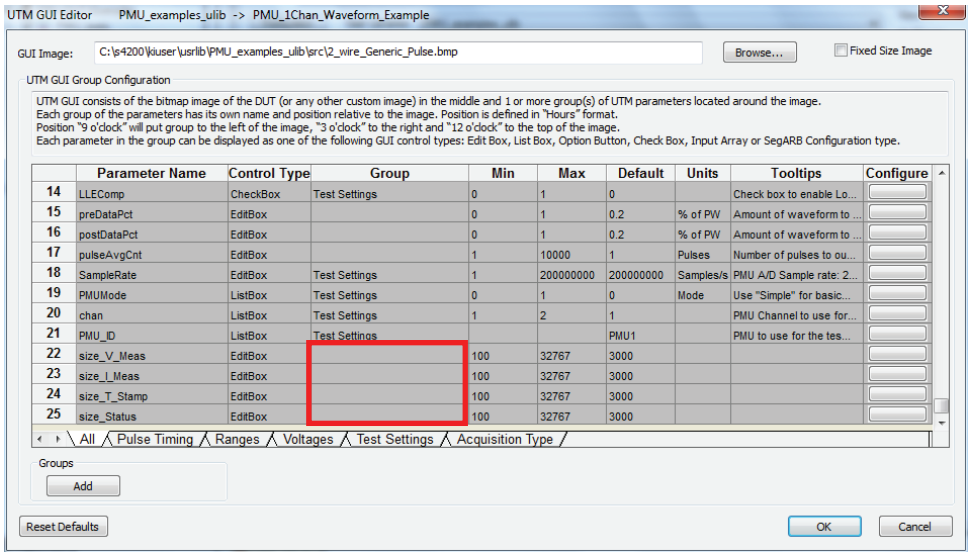
The maximum number of parameters in a group is ten (10). Limit the number of parameters in a group so that it displays completely (group boxes do not have scroll bars). Generally, it is better to reduce the number of displayed parameters so all groups can be expanded at the same time.

When creating a user UTM GUI, keep the following items in mind:

- A UTM GUI view definition must have at least one group.
- Each group must have a unique name.
- Note that the "All tab" displays a special type of group; this group is not displayed on the UTM GUI view.
- For a test parameter to appear in the GUI view, it must be in a group (this is set in the UTM Parameter GUI Configuration dialog box).
- Display only the important or commonly changed parameters; a GUI with fewer parameters is easier to understand and use than one with too many parameters.

All parameters for a test do not have to be placed in a group; only place those parameters that you want to display to the user. For instance, for the majority of tests, the size values of the output arrays can be left at the default values and do not have to be displayed. Figure 8-62 shows that the four size parameters (`size_V_Meas`, `size_I_Meas`, `size_T_Stamp`, `size_Status` in rows 22-25) do not have a Group and will not be shown in the GUI view. Other parameters may also be left out of the view, depending on the typical use of the user module. The unassigned parameters will not be shown in the GUI view but are still available and active. Use the Classic view (Figure 8-55) to view or adjust parameters that are not in the GUI view.

Figure 8-62
Size parameters not assigned a group (not displayed in GUI view)

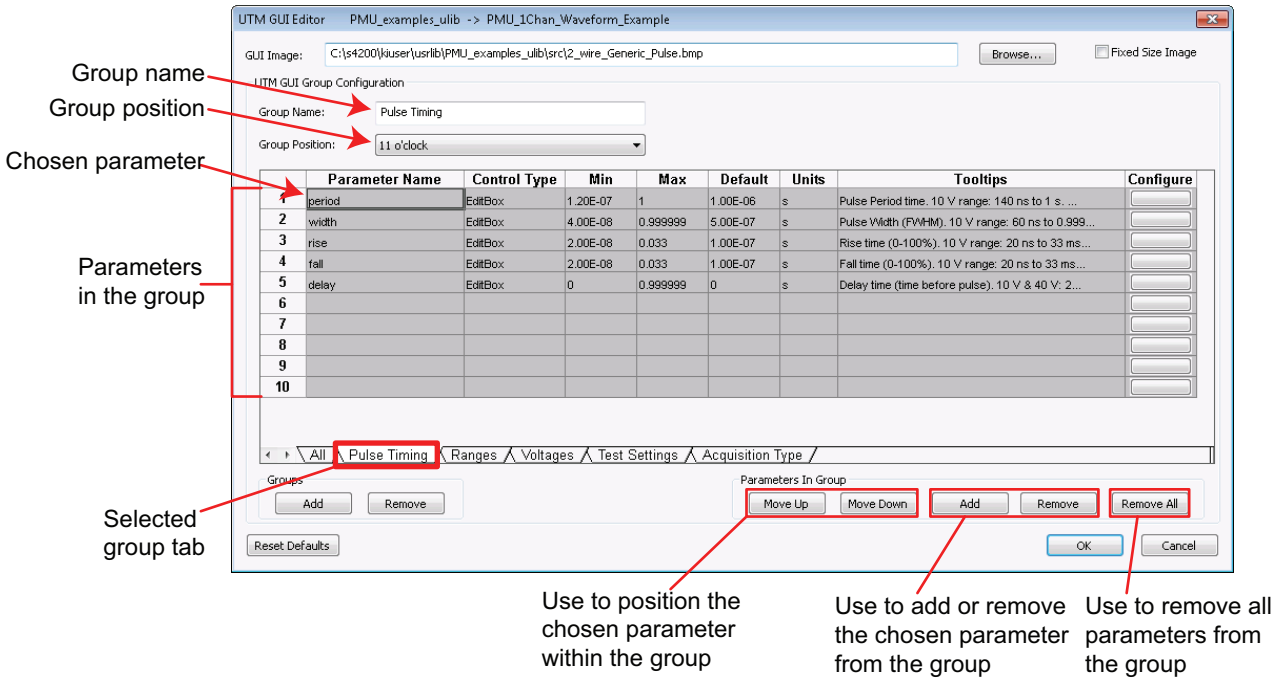


To add a group

1. Open the main UTM GUI editor (see “Example of using the editor” on page 8-65).
2. Click the **Add** button (this button is located towards the lower left corner of the Main UTM GUI Editor, see Figure 8-62).
3. Configure the group (fill-in the group name, select the group position, add and configure the parameters).
4. Click the **OK** button to save any changes; click the **Cancel** button to exit the dialog without saving any of the changes.

The view of a group tab (see the "Pulse Timing" tab in Figure 8-63) is a subset of the All tab (Figure 8-62). The differences between the two include the group name, the group position, and the parameter list (the group tab's parameter list contains only the parameters in the group, whereas the UTM's All tab contains all parameters).

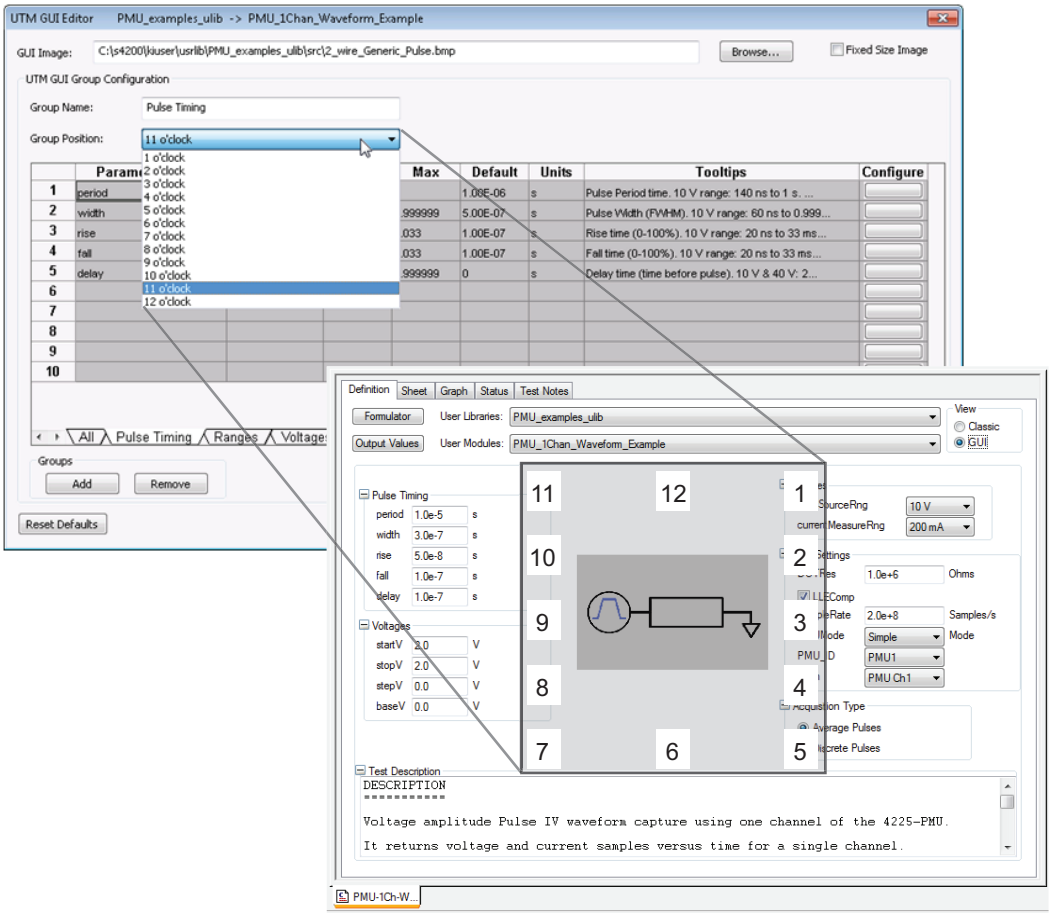
Figure 8-63
Example group view for pulse timing



To modify a group

1. Click the group's tab (a tab for each group located in either editor). If a tab is not shown, use the left or right sheet buttons to move the groups until the tab displays (the arrow buttons are located to the left of the All group tab).
2. The following group-level items can be modified (see the "Pulse Timing" tab in Figure 8-63):
   - **Group name**: The text box is case sensitive. The group name entered in this view will appear exactly as entered. Avoid using non-text characters, such as LF (Line feed) or CR (carriage Return), in this field. Use standard characters, a-z, A-Z, 0-9, and space.
   - **Group position**: Set each group's position with respect to the GUI Image bitmap by selecting a clock hour from the Group Position list (see Figure 8-64). This sets the relative position; the number of parameters in each group defines the final layout. For example, if two groups next to each other have a lower number of parameters, the groups will have more space between them. On the Model 4200-SCS display, there is limited space at 12 o'clock (above the image) and 6 o'clock (below the image). Only two or three parameters typically fit in this space, but this number varies based on the GUI image size. If you place a group with a large number of parameters in the 12 o'clock or 6 o'clock positions, it will not open; change the group's position or reduce the number of parameters.
   - **Parameter order**: Select a parameter row, and then click the Move Up or Move down button to change the parameter's position in the group box.
   - **Parameters in the group**: To remove a parameter from a group, select the parameter row and click the **Remove** button (to removal all parameters from the group, click the **Remove All** button). To add a parameter, click the **Add** button and then select and configure the parameter.

Figure 8-64
UTM GUI Editor group position example



Selecting a GUI image

NOTE *Some items in the UTM GUI Editor can make GUI level modifications; if you change any of these items, you will change them for the entire UTM GUI. These items include GUI Image and Fixed Size Image.*

Use the main GUI Editor dialog (Figure 8-59) to select a bitmap image. Every UTM GUI must have only one bitmap image. By default, the image is taken from the device plan where the UTM resides. In Figure 8-55 and Figure 8-56, the Device Plan for this example is `4terminal-n-fet`. The file path for the default image is shown on new GUIs. The image must be a bitmap image and the size should be about 120 x 100 pixels up to about 480 x 400 pixels and a file size < 500 kilobyte. Full color bitmaps may be used. Larger pixel size images render better on the Model 4200-SCS screen.

Fixed size image

As the user changes the size of the KITE window, the bitmap image is scaled. Any text or fine detail in the image may be distorted at certain window sizes. To prevent scaling of the image, select **Fixed Size Image** (a check box located in the upper right corner of the main UTM GUI Editor window, Figure 8-56). When using a fixed image size, a smaller pixel size image may be necessary for large numbers of test parameters or groups. The default KITE images are stored in

the source directory of the user library. The example above (Figure 8-61, Figure 8-62, Figure 8-63, and Figure 8-64) is part of user library PMU_examples_ulib, so the bitmap files are located in `c:\s4200\kiuser\usrlib\PMU_examples_ulib\src`.

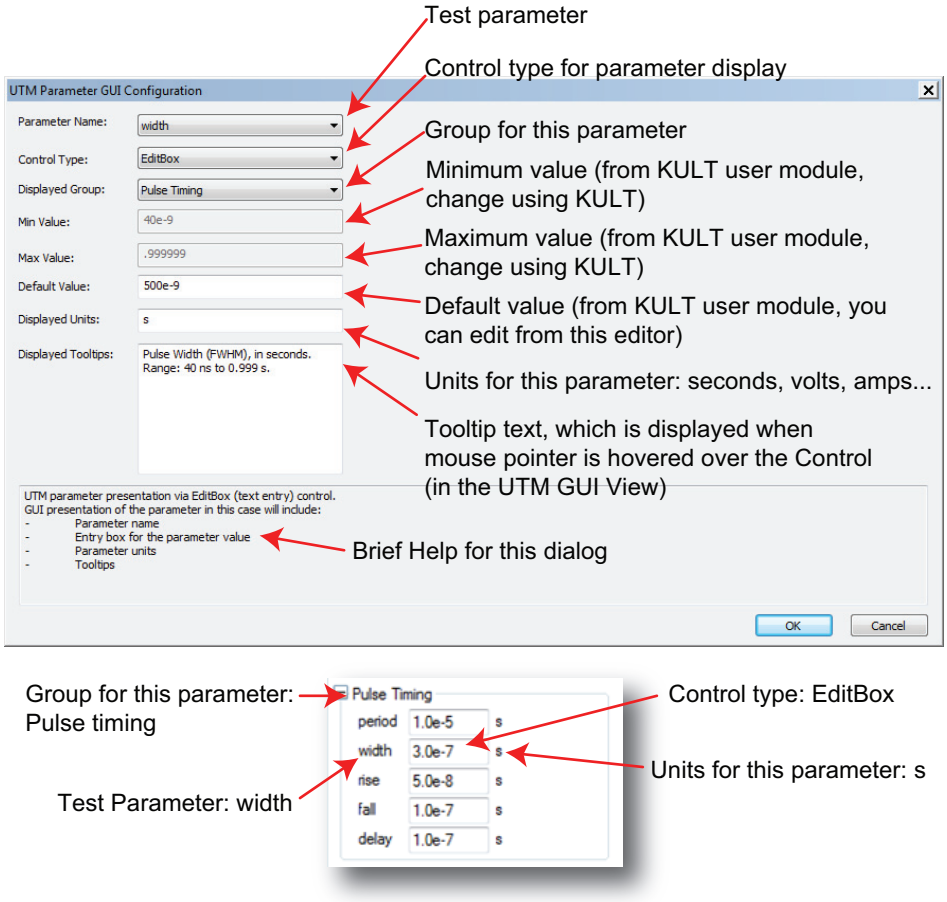
NOTE *Some items in the UTM GUI Editor can make GUI level modifications; if you change any of these items, you will change them for the entire UTM GUI. These items include GUI Image and Fixed Size Image.*

Editing the attributes for a test parameter

The entire main GUI Editor table is a summary (not editable). To edit a test parameter's attributes, open the UTM Parameter GUI Configuration dialog by double-clicking anywhere on the row or by clicking the Configure button (located at the right end of each row). The content of the configuration dialog is different for each type of control. You can open the UTM Parameter GUI Configuration dialog and configure a parameter's attributes using the All tab, or in the specific group's tab (if the parameter is assigned to a group). In Figure 8-62, there are five group tabs shown: Pulse Timing, Ranges, Voltages, Test Settings, and Acquisition Type, in addition to the All tab.

Figure 8-65 illustrates a sample UTM Parameter GUI Configuration dialog. This specific instance illustrates the width parameter (the Parameter Name is set to `width`). The parameters that are available for configuration depend on which tab was used to enter the parameter configuration editor. If the configuration dialog is accessed from the All tab, all the parameter names will be available in the related menu; if accessed from a group's tab, only the parameter names available in the group will be available.

Figure 8-65
GUI Configuration for the width parameter (EditBox)



Control type

Using the width test parameter as an example, the type of control attribute is the first configurable attribute (Control Type). In Figure 8-65, the control type is set to EditBox. The control type selected dictates the content and layout of the UTM Parameter GUI Configuration dialog. See "Control types" on page 8-71 for additional information.

Displayed group

The next configurable item in the width test parameter is the "Displayed Group." In Figure 8-65, the width parameter belongs to the "Pulse Timing" group. In the UTM GUI, this parameter appears in the "Pulse Timing" group box. With this setting, you can this access this parameter through the "Pulse Timing" or "All" tabs.

Minimum, maximum, and default value fields

These parameters are the limits and defaults for the parameter and are available as integer and double types. Typically, these three values are defined in the user module using KULT. If the minimum and maximum values are defined in the user module, you can only modify the default value from within the UTM Parameter GUI Configuration dialog. If the minimum and maximum values are not defined in the user module, you can edit them using the UTM Parameter GUI Configuration dialog.

Displayed units field

Changing the units text entry field changes the displayed units of measurement, but not the value (no conversions are made); make sure to use the same units as the applicable command (see the specific command for details). As a quick check, you can make sure the units reflect the same units of measure used in the "Min Value" and "Max Value" fields.

Displayed tooltips field

This field is a text entry fields. Avoid using non-text characters, such as LF (Line feed) or CR (carriage Return), in this field. Use standard characters (a-z, A-Z, 0-9, and space) and no symbols. Generally, keep tooltips short (over two words and usually less than eight), use simple present tense, always use clear and consistent language, and check your spelling.

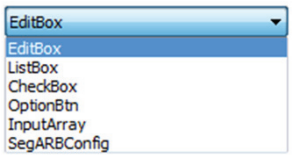
Help

If any help concerning this parameter is available, it will appear at the bottom of the configuration dialog.

Control types

You can set the control type to one of six different types of control types available for entry and display of a parameter in the UTM GUI view (see Figure 8-66).

Figure 8-66
Parameter control types



EditBox

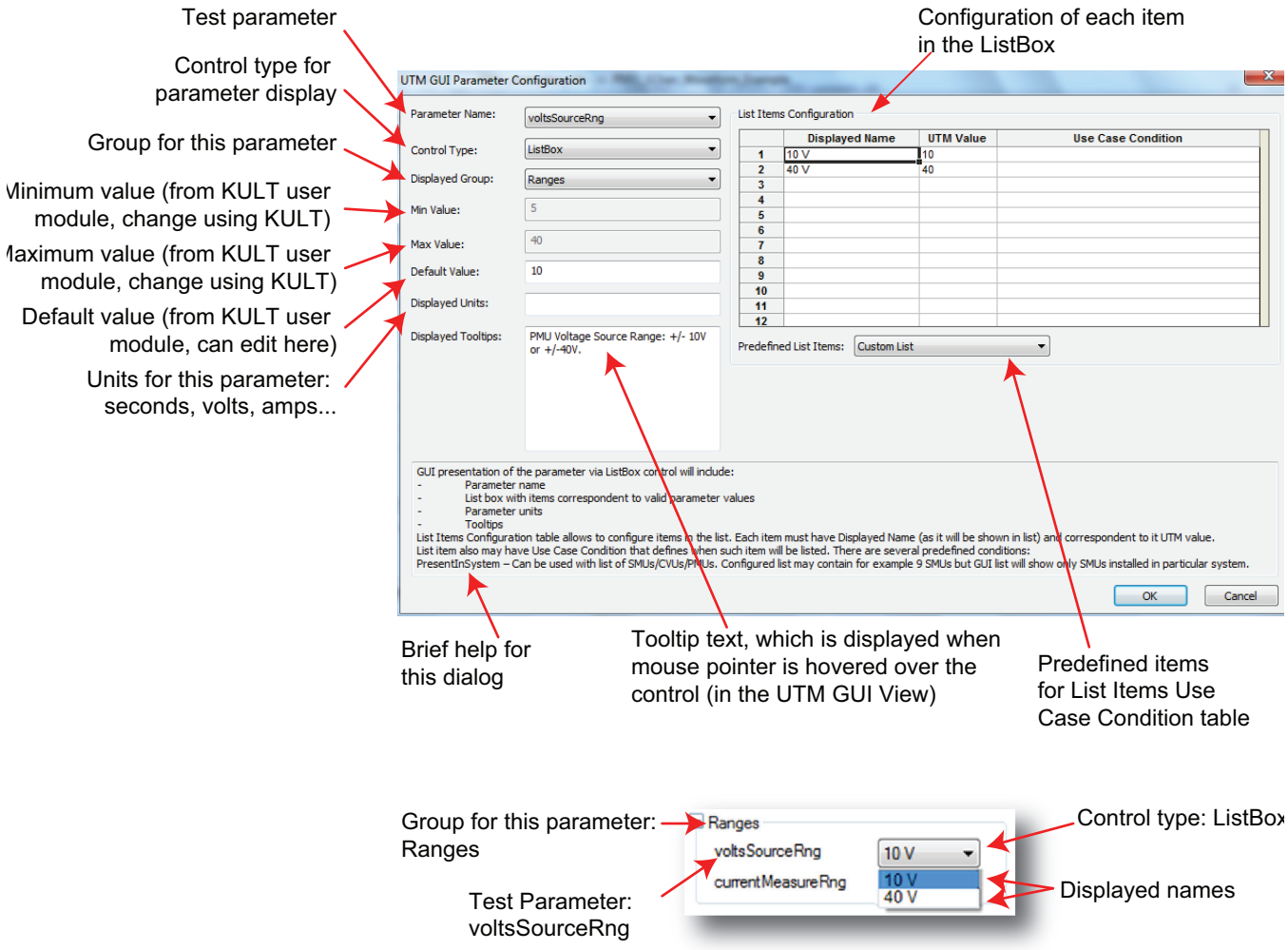
The EditBox Control type (shown in Figure 8-65) is the simplest method to allow changing of a scalar parameter value. You can use this control type for source values (such as voltage or current), or pulse timing parameters, anywhere a wide range of continuous values is needed. This control type may be used for all non-array inputs. It is also the default control type for all non-array inputs in dynamically generated UTM GUI views (example of dynamically generated GUI view in Figure 8-57).

ListBox

Use a list box to specify a value that the user can select. Choosing this type of control places limits on allowable values and makes use of the module more efficient. Measure and source ranges are parameters that usually benefit from choosing a list box. Configuring a list box control requires additional information about the parameter when it is created, but eases the learning curve for the use of a user module.

This control provides configuration of parameters that have a limited number of options or states. A ListBox can hold a minimum of 2, to a maximum of 12 values. Figure 8-67 shows the ListBox dialog for the voltsSourceRng parameter. This parameter has only two values: a number 10 and a number 40.

Figure 8-67
GUI Configuration for the voltsSourceRng parameter (ListBox)

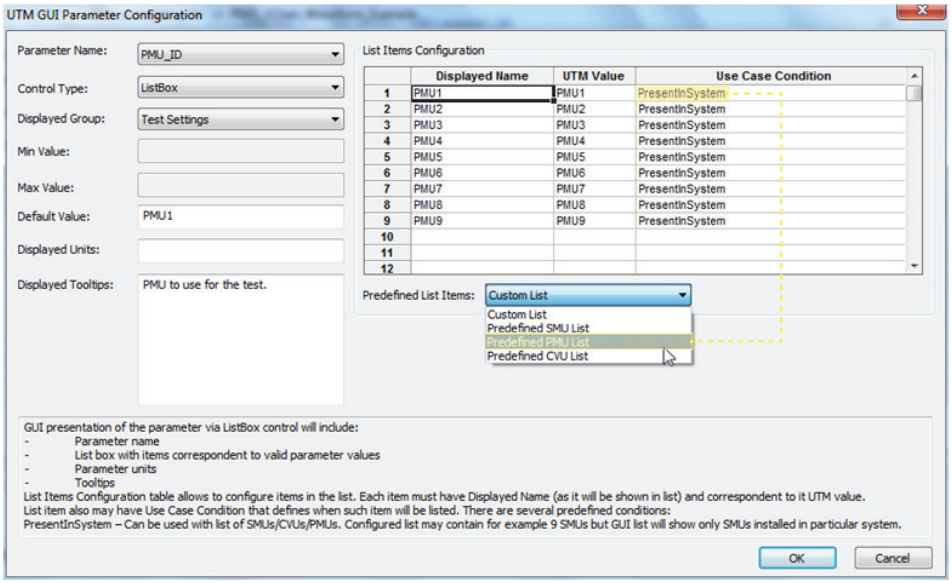


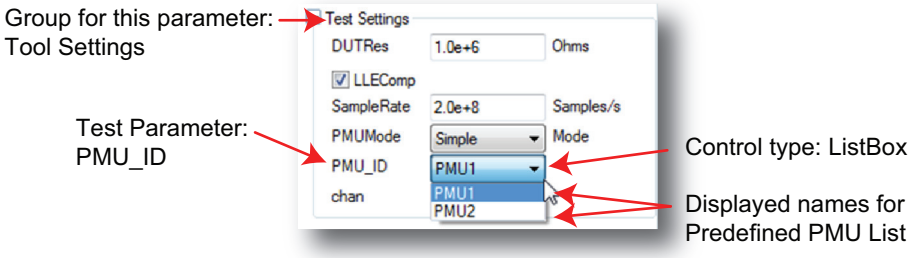
If a ListBox Control type is chosen for a parameter, fill in at least one row of the List Items Configuration. Make sure to fill out both the Displayed Name and UTM Value. For the Displayed Name, choose one that briefly explains or represents the UTM value. Create short Displayed names (one or two words are best). If the user module includes a default value for this parameter, then Default Value field will be populated but can be changed using this dialog. Also make sure to enter appropriate Displayed Units, if applicable. Note that "Displayed Units" field does not affect the test or parameters. You can use it to assist the user in understanding the parameter values. Enter text in the "Displayed Tooltip" field with a short, informative phrase or sentence (see Displayed tooltips field for more information). When finished, click the OK button to exit this dialog (click the Cancel button to leave without saving).

The right-most column "Use Case Conditions," allows for conditionally controlled display of the list box choices. If the Use Case Condition is true, the Displayed Name for the row appears in the ListBox. For example, if a particular current range is only available when a Model 4225-RPM (or SMU PreAmp) is connected on the chosen channel, you can add logic to permit these use case conditions. For the instrument cards, the choices available in the drop down are shown in Figure 8-68. The entries shown in the table were the direct result of choosing the "Predefined PMU List" in the dropdown. These predefined options will populate the table with the chosen card for slots 1-9. The key is that the "PresentInSystem" condition means that the dropdown for the card will only display card options that are installed in the Model 4200-SCS. Figure 8-68 shows the PMU_ID

dropdown in a system with two Model 4225-PMU cards. SMU lists, CVU lists, and a customizable list are also available as items in the Predefined List Items list.

Figure 8-68
Figure 8-14L. ListBox UTM Parameter GUI Configuration for PMU_ID





In addition to the predefined list items, you can create other conditions to simplify the use of the user module or reduce the possibility of errors. Before the test runs, a user module and UTM has no information about the system. Therefore, errors must be generated by manually querying the hardware and trying out a specific command and retrieving the error status. The Use Case Conditions in the UTM GUI view has information about the system configuration. This can be utilized to direct the UTM GUI user to selection of appropriate parameter values. Figure 8-69 shows the UTM GUI Parameter Configuration dialog for the parameter currentMeasureRng.

The Use Case Condition field conditionally displays the Displayed Name in the list box. The expression you supply for this field must evaluate to True or False. If blank, the condition is evaluated as True. In other words, this field is evaluated as: IF the Use Case Condition = True, THEN show the Displayed Name in ListBox. For example, in Figure 8-69, the first line's Use Case Condition means, "If voltsSourceRng is equal to 40, then display "800 mA" in the drop-down list. This effectively allows the 800 mA range to display and be selected when the voltage range is set to 40 V (see Figure 8-70).

Note the complexity of the currentMeasureRdg parameter's lower-current use case conditions. These use case conditions account for the three different sets of the current measure ranges which depend on the voltage range selected (10 V or 40 V) and the presence of a Model 4225-RPM (which adds the lower current measure ranges to the 10 V range). For this

parameter, the content of the list box is described Table 8-7. Table 8-8 lists keywords and operators available for use in the Use Case Condition field.

Figure 8-69
ListBox UTM Parameter GUI Configuration for currentMeasureRng

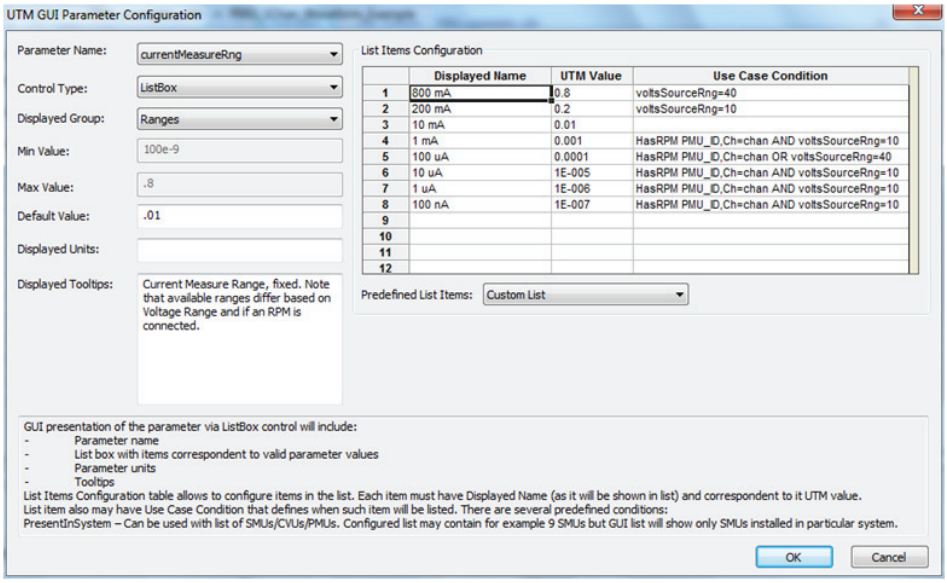
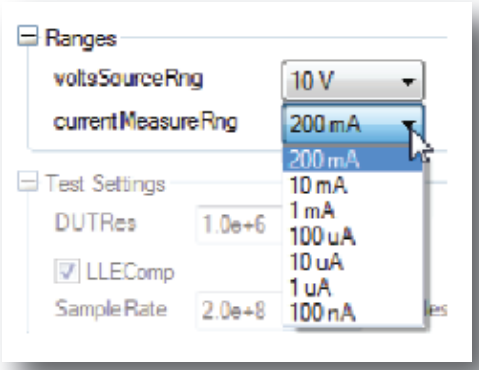


Figure 8-70
currentMeasureRng list box with and without a connected RPM

With Model 4225-RPM connected



With no Model 4225-RPM connected (PMU measure ranges only)

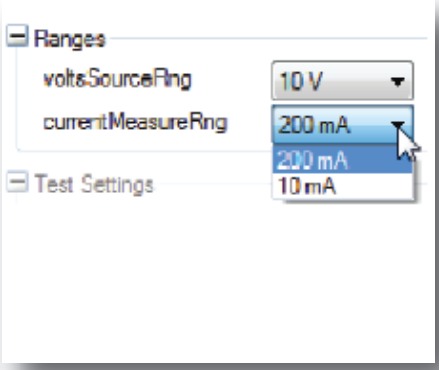


Table 8-7
Description of "Use Case Conditions" in currentMeasureRng example

	Displayed Name	UTM Value	Use Case Condition	Comment
1	800 mA	0.8	`voltsSourceRng=40`	Display this name when the chosen voltage range = 40 V[1]
2	200 mA	0.2	`voltsSourceRng=10`	Display this name when the chosen voltage range = 10 V

Table 8-7 (continued)
Description of "Use Case Conditions" in currentMeasureRng example

	Displayed Name	UTM Value	Use Case Condition	Comment
3	10 mA	0.01		Always display, as the 10 V, 40 V and RPM have a 10 mA measure range
4	1 mA	0.001	`HasRPM PMU_ID,Ch=chan AND voltsSourceRng=10`	Display this name only when the voltage range is 10 V and there is an RPM on the chosen channel (variable name = chan).
5	100 μA	0.0001	`HasRPM PMU_ID,Ch=chan OR voltsSourceRng=40`	Display this name in 2 cases: if there is an RPM or the voltage range is 40 V.[1]
6	10 μA	1E-005	`HasRPM PMU_ID,Ch=chan AND voltsSourceRng=10`	Display this name only when the voltage range is 10 V and there is an RPM on the chosen channel (variable name = chan).
7	1 μA	1E-006	`HasRPM PMU_ID,Ch=chan AND voltsSourceRng=10`	Display this name only when the voltage range is 10 V and there is an RPM on the chosen channel (variable name = chan).
8	100 ηA	1E-007	`HasRPM PMU_ID,Ch=chan AND voltsSourceRng=10`	Display this name only when the voltage range is 10 V and there is an RPM on the chosen channel (variable name = chan).

1.The 40 V range has 800 mA, 10 mA, and 100 μA current measure ranges.

Table 8-8
ListBox "Use Case Condition" keywords and operators

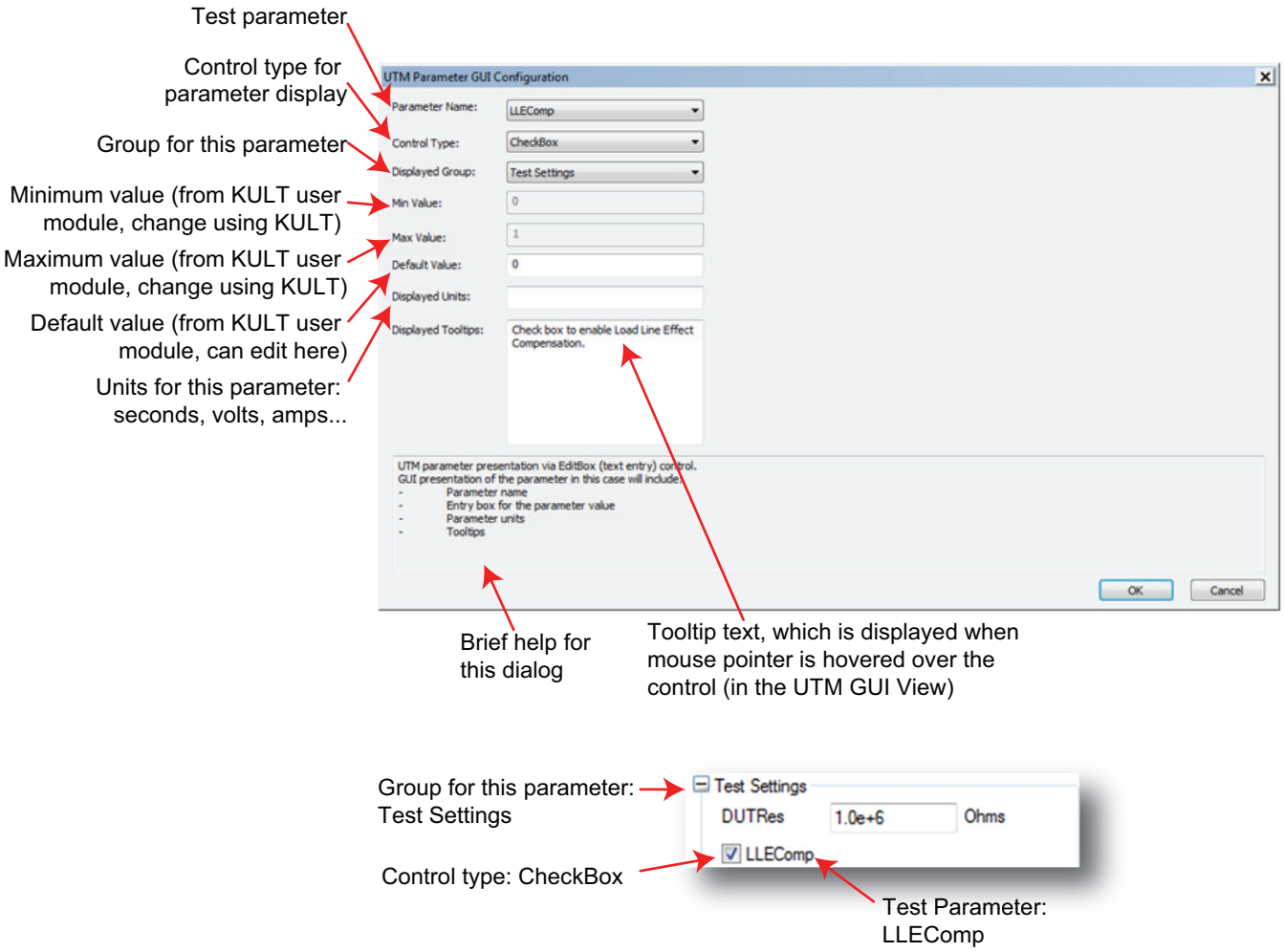
Keyword or operator	Explanation	Comment
`PresentInSystem`	Used for 4200-SCS instrument card: SMU, CVU, PMU, PGU	This condition must be alone in the Use Case Condition field. It cannot be combined with other conditions or operators listed below.
`HasPA` *smuid*	True if SMU Preamp is connected to *smuid*	Note that *smuid* can be a constant, "SMU1," or a parameter name.
`HasRPM` *pmuid*, `Ch=`*chanid*	True if RPM is connected to *chanid* of *pmuid*.	Both *pmuid* and *chanid* may be constants (PMU1, 1) or parameter name (PMU_ID, chan). Follow spacing as shown.
`AND, OR`	Logical operators	Separate conditions with a space before and after each operator. Maximum of two operators for each row.
`=, !=, <, >, >=, <=`	Comparison operators	Can compare constants or values of parameters. If the argument to the right of the operator is a parameter, it must be enclosed in curly brackets {}. Example 1: VrangeCh1 = 10 Example 2: VrangeCh1 = {VrangeCh2}

CheckBox control

Use a check box control (CheckBox) for parameters that have two values or states. This control returns a zero (0) or a one (1) to the user module, representing whether the check box is selected (1) or not (0). As an example, the LLEComp parameter from the PMU_1Chan_Waveform_Example user module has just two states (Figure 8-60): Disabled and Enabled. These values refer to the state of the LLEC compensation (see "Load line effect compensation (LLEC) for the PMU" on page 16-60 of the Reference Manual) and is used in the

LPT command pulse_meas_wfm (see "pulse_meas_wfm" on page 8-188). This is the simplest control in the GUI Editor; there is not much to configure (Figure 8-71). Provide or change the Default Value and supply explanatory Tooltip text. As the actual parameter name is used as a label, make sure to provide a tooltip that removes any ambiguity as to the meaning of the checkbox. If desired, you can also note the unit of measurement in the Displayed Units field (this is for reference of the UTM GUI programmer). When finished, click the **OK** button to exit this dialog (click the **Cancel** button to leave without saving).

Figure 8-71
CheckBox UTM parameter GUI configuration



OptionBtn control

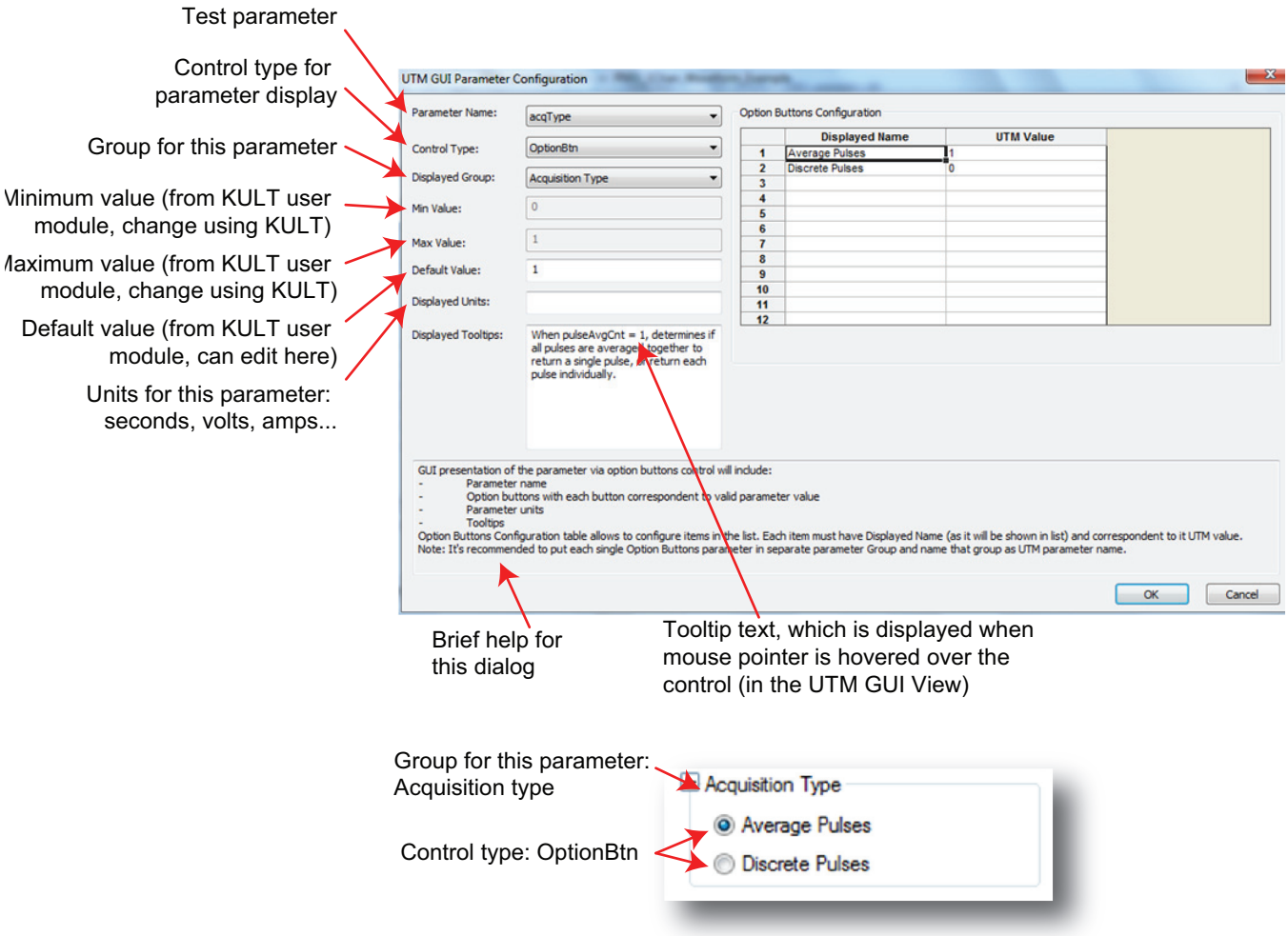
Use the option button control (OptionBtn, which can also be referred to as a radio button or radio button control) when only one item of a group may be selected. This control is also useful for a parameter with a limited number of values (at least two), up to a maximum of four values. Consider using a list box for parameters requiring additional values (a list box can hold from 2 to 12 values).

NOTE *List boxes take up less space in the GUI View because only one state displays unless being selected; the option button control must show all choices.*

This control must be used within its own group. See an example of this control in the lower right of Figure 8-60 for the AcqType parameter (the Acquisition Type group is an option button control). The configuration of an option button (Figure 8-72) is similar to a list box. Note that this parameter could also be implemented with a check box control or a list box control. Either would require less space on the UTM GUI, but using an option button control for this parameter promotes the either-or nature of this parameter. Using an option button permits different values to be returned for each choice in the same way a list box; a check box control only returns a zero (0) or a one (1).

In this example, these values correspond to the two measurement modes for the spot mean measurement LPT command pulse_meas_wfm (see "pulse_meas_wfm" on page 8-188). The two measurement modes are Discrete and Average. This option determines whether measurements from multiple pulses (set by pulseAveCnt parameter) are to be averaged together into a single value (average), or as each pulse with its own measurement (discrete). Refer to "Waveform measurements" on page 8-192 for additional information on this measurement mode.

Figure 8-72
OptionBtn UTM parameter GUI configuration



InputArray control

Use the input array control (InputArray) for array input types in user modules (Figure 8-73). Configure the parameter variable types using the Parameters tab in KULT (see "Parameters tab area" on page 8-7).

For this control type, the example uses the vlfcv_measure_sweep_freq user module located in the VLowFreqCV user library. When placed in the UTM GUI, the user can click the Edit button and enter array values (see Figure 8-73).

NOTE *Make sure the values the user enters in the array tables are contiguous (no blank rows in the middle of filled cells). An error results if one (or more) blank row is in the table before the end of the data.*

The configuration of an input array control (Figure 8-74) is similar to an edit box. The user module does not provide support for the minimum, maximum, or default values for the arrays; only the UTM GUI provides this capability. The minimum and maximum values are single values that provide bounds for each entry. The default values are a series of values for the array. Enter the default values for the array separated by commas only (do not use any spaces).

Figure 8-73
UTM GUI View InputArray control

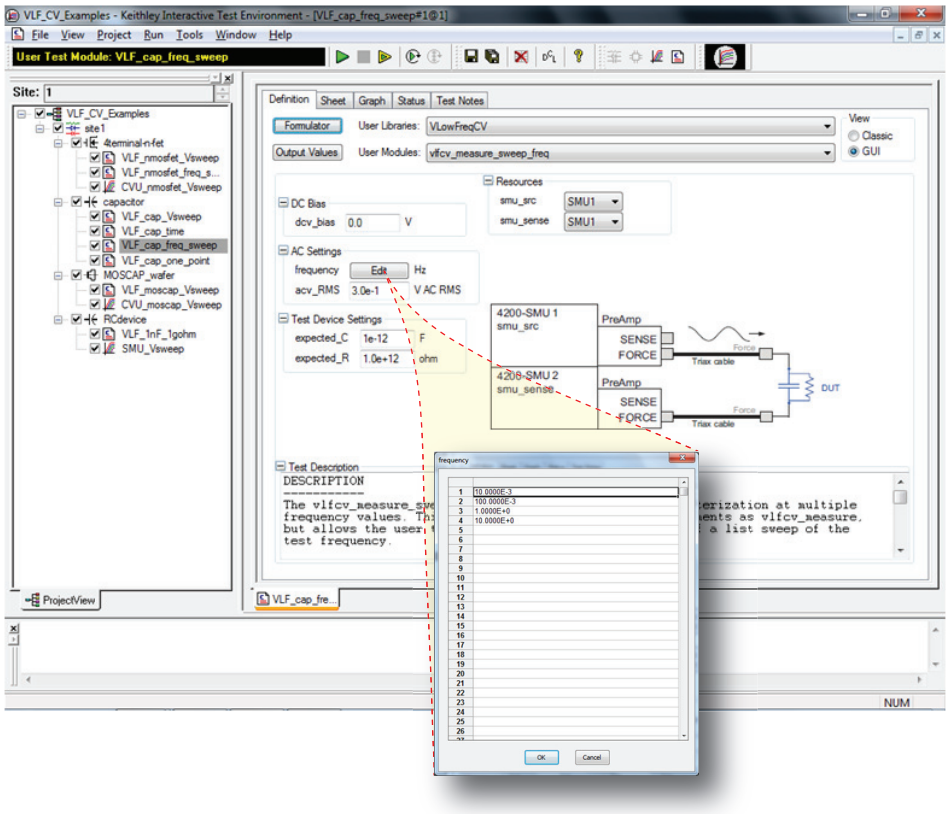
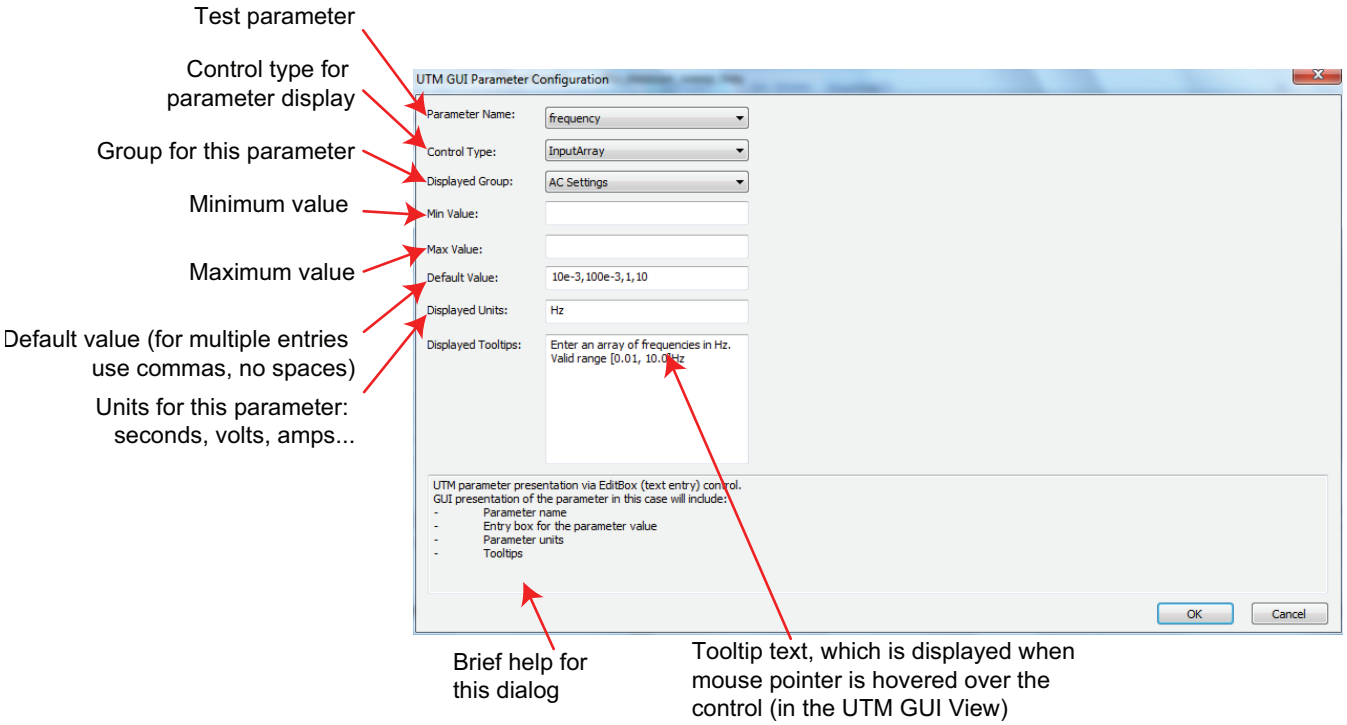
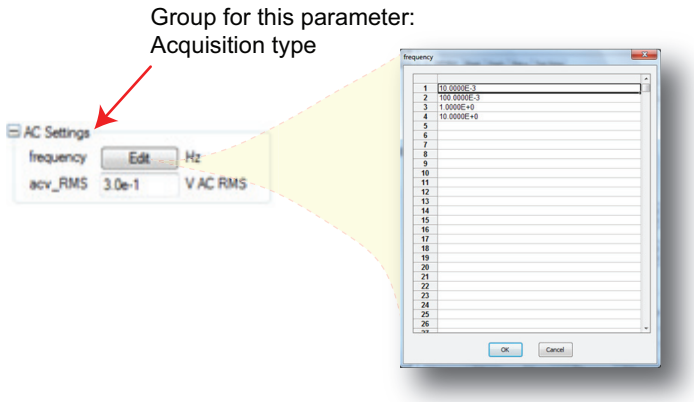


Figure 8-74
InputArray UTM parameter GUI configuration



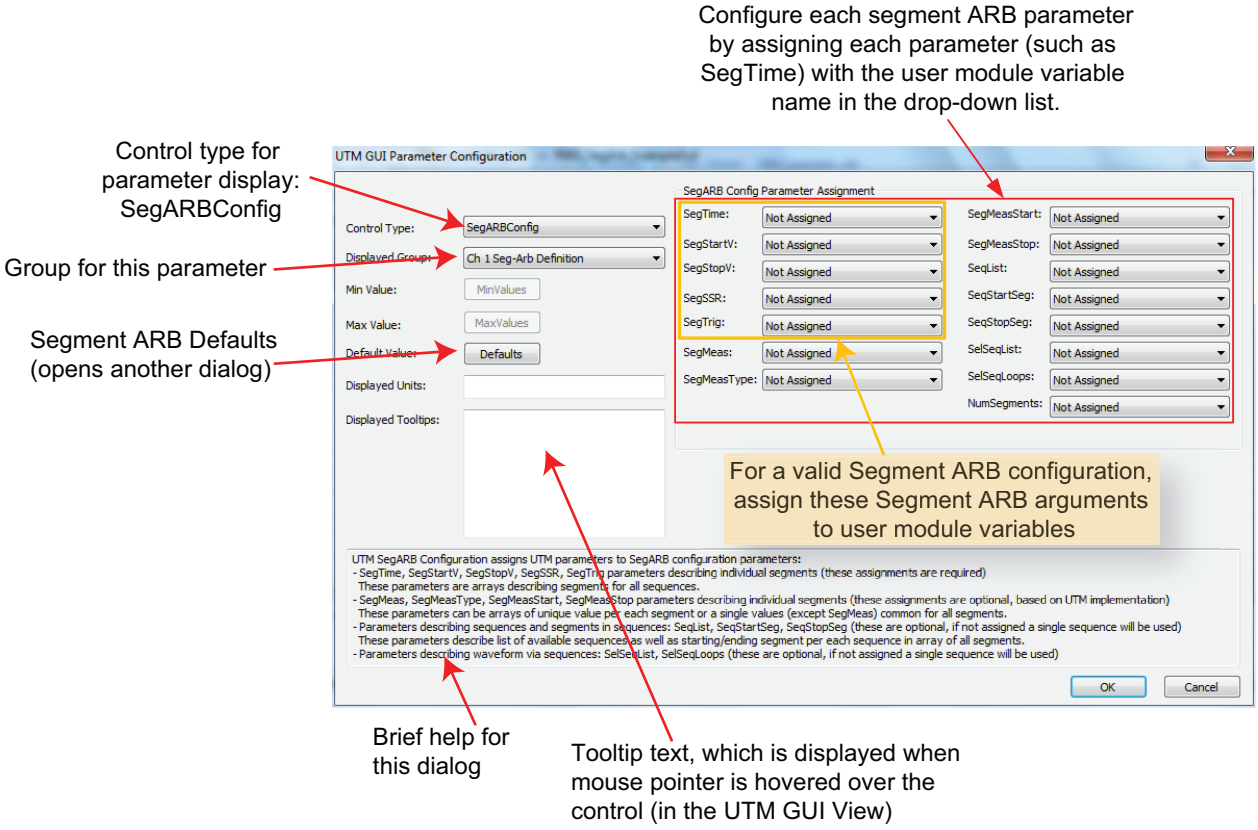


SegARBConfig

The SegARBConfig is the most complex control type available for the UTM GUI view. The Segment ARB® capability has many parameters, with most of them in arrays. This control type provides the interface to user modules making use of two specific LPT commands: “seg_arb_sequence” on page 8-204 and “seg_arb_waveform” on page 8-206. There are two dialogs that configure the Segment ARB GUI view:

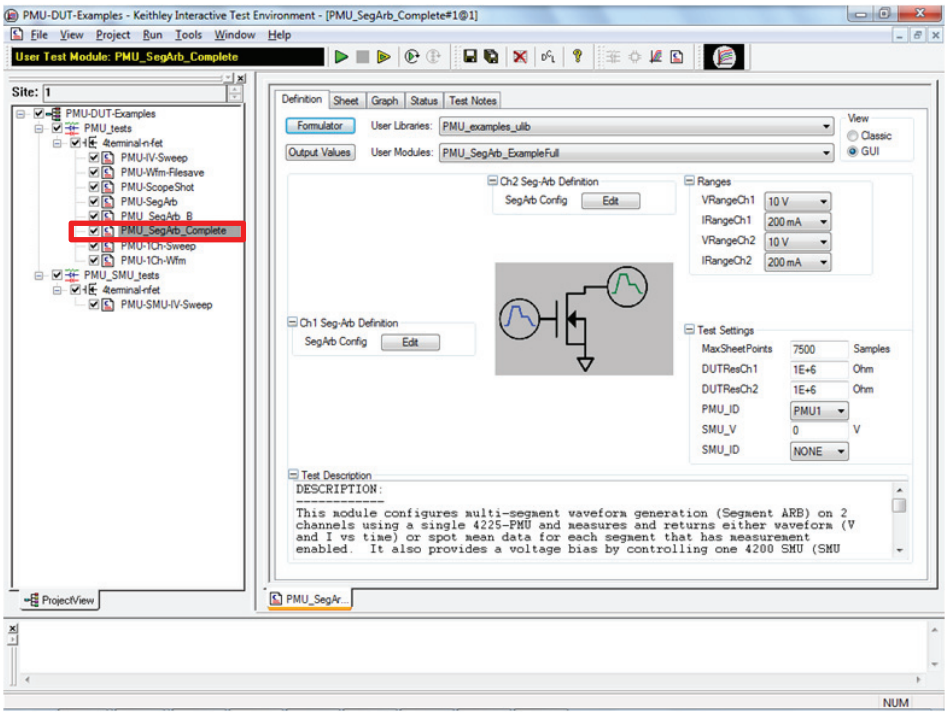
- "UTM GUI Parameter Configuration dialog (shown Figure 8-75). This dialog is the primary dialog that you can use to configure the SegARBConfig control.
- "Segment ARB Defaults Configuration dialog (shown in Figure 8-83). This is the second dialog that you can use to configure the channel's sweep values.

Figure 8-75
SegARBConfig UTM GUI Parameter Configuration dialog



For this control type, the PMU_SegArb_ExampleFull user module is used in the example (see Figure 8-14R). This module is included in the PMU-DUT-Examples project (in the _Pulse project folder) as the UTM PMU_SegArb_Complete (Figure 8-76). Figure 8-76 shows the Segment ARB configuration control as the only control in the group. You can add other controls, but there can be only one Segment ARB control in a group.

Figure 8-76
UTM GUI View for the SegARBConfig



In addition to the settings that are configured like the other control types (such as displayed group, displayed units, displayed tooltip text), the SegARBConfig control requires assignment of multiple parameters. Each Segment ARB argument requires association with the corresponding variable name in the user module (Figure 8-75 and Figure 8-77). Also, each parameter assignment involves many parameters (red box, Figure 8-76). This is in contrast to other controls; the InputArray control has a single parameter in its parameter configuration dialog.

Figure 8-77
SegARBConfig Parameter Configuration listing variables

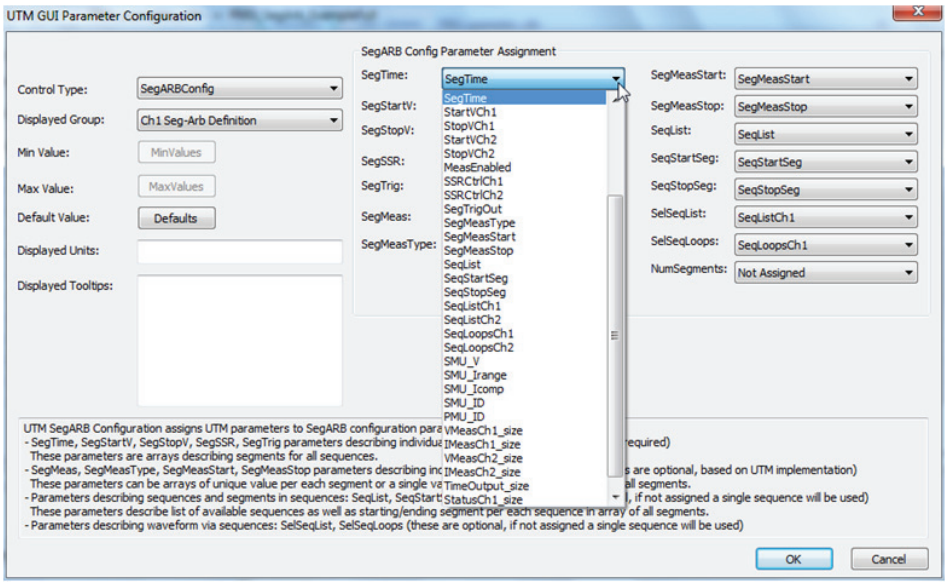


Table 8-9 summarizes all the parameters in the SegARB Config control. In the table, each parameter is shown mapped to its parameter target. As mentioned above, the Segment ARB functionality is primarily contained in two LPT commands: "seg_arb_sequence" on page 8-204, "seg_arb_waveform" on page 8-206. In addition, a few parameters are unique to the SegARBConfig control and are explained later in this section.

Table 8-9
SegARBConfig Control Parameters

SegARBConfig parameter name	Parameter target (LPT function or SegARBConfig)	Variable name in function (parameter target)	Type	Required[5]	Description
SegTime	seg_arb_sequence	Time[1]	Double array	Yes	Time duration for each segment
SegStartV	seg_arb_sequence	StartV[1]	Double array	Yes	Segment start voltage
SegStopV	seg_arb_sequence	StopV[1]	Double array	Yes	Segment stop voltage
SegSSR	seg_arb_sequence	SSR[1]	Integer array	Yes	Solid State Relay control, per segment
SegTrig	seg_arb_sequence	Trig[1]	Integer array	Yes	Trigger output state for each segment
SegMeas	SegARBConfig	SegMeas[3]	Integer array	No	Measurement Enabled[6]
SegMeasType	seg_arb_sequence	MeasType[1]	Integer array or Integer[4]	No	Measurement Type: None, spot mean, or waveform PULSE_MEAS_NONE (0) PULSE_MEAS_SMEAN_PER (1) PULSE_MEAS_WFM_PER (2)
SegMeasStart	seg_arb_sequence	MeasStart[1]	Double array or Double[4]	No	Start point for measurement window, for each segment. From 0 to 1 (100%).
SegMeasStop	seg_arb_sequence	MeasStop[1]	Double array or Double[4]	No	Stop point for measurement window, per segment. From 0 to 1 (100%).
SeqList	SegARBConfig	SeqList[3]	Integer array	No	List of defined sequences in SegARBConfig[3]
SeqStartSeg	SegARBConfig	SeqStartSeg[3]	Integer array	No	Start segment array value in SegARBConfig[3]
SeqStopSeg	SegARBConfig	SeqStopSeg[3]	Integer array	No	Stop segment array value in SegARBConfig[3]
SelSeqList	seg_arb_waveform	Seq[2]	Integer array	No	List of sequences that define the Seg-Arb waveform.
SelSeqLoops	seg_arb_waveform	SeqLoopCount[2]	Integer array	No	Array of loops values for each sequence in the Seg-Arb waveform.
NumSegments	seg_arb_sequence	NumSegments[1]	Integer	No	Number of segments in a sequence.

1. Variable name in "seg_arb_sequence" on page 8-204.
2. Variable name in "seg_arb_waveform" on page 8-206.
3. Variable name from SegARBConfig.
4. If using a single value instead of an array, the SegARBConfig control will automatically assign the value to each segment.
5. Required: Yes, required in SegARBConfig; No, not required in SegARBConfig.
6. SegMeas parameter turns a measurement on or off for each segment and is independent of the SegMeasType (based on implementation in user module).

Except for the `NumSegments` parameter, the SegARB Config Parameter Assignment parameters are arrays (see the red box in Figure 8-75). `SegMeasType`, `MeasStart`, and `MeasStop` can be set to either a single value (either integer or double, depending on the parameter) or as an array. If

a parameter is configured as a single value input by the user module, then the UTM GUI will display it as a ListBox (see red box in Figure 8-78). You must associate each Segment ARB parameter (in the orange box shown in Figure 8-75) with the appropriate variables in the user module. If not, an error will result.

The other parameters (outside of the box) are optional and based on the implementation of the Segment ARB capability within a particular user module. Optional parameters meaning that the user of the test is not required to specify them; these parameters must still be set in the user module to create a valid Segment ARB test.

Multi-sequence tests

Three parameters, although not required for the SegARBConfig control, are required for multi-sequence tests. These three parameters are `SeqList`, `SeqStartSeg`, and `SeqStopSeg`. They are used both by the SegARBConfig control and the user module to define the multiple sequences using the "seg_arb_sequence" on page 8-204 function and also the multi-sequence waveform using the "seg_arb_waveform" on page 8-206 function. These sequencing parameters allow data of each sequence to be stored within a single array for each parameter.

The StartVCh1 array in the user module PMU_SegArb_ExampleFull illustrates the use of the three-parameter arrays. In the array, assume there are two sequences: sequence one having nine segments and sequence two having seven segments. For the first sequence, the value for `SeqStartSeg` is 1 and for `SeqStopSeg` is 9. For the second sequence, the value for `SeqStartSeg` is 10 and `SeqStopSeg` is 16 (see Figure 8-80). This mapping applies to all array-parameters in the "seg_arb_sequence" on page 8-204. It is then utilized by the code in PMU_SegArb_ExampleFull to define each sequence by using the indices to pass the appropriate values for each array in each sequence. The code in this user module loops through the rows shown on the left in Figure 8-80, defining each sequence by calling `seg_arb_sequence`.

NOTE *Make sure all values in the Segment ARB array tables are contiguous (no blank rows between filled cells). An error results if one (or more) blank row is in the table before the end of the data.*

Figure 8-78
Segment ARB GUI UTM configuration for Channel 1

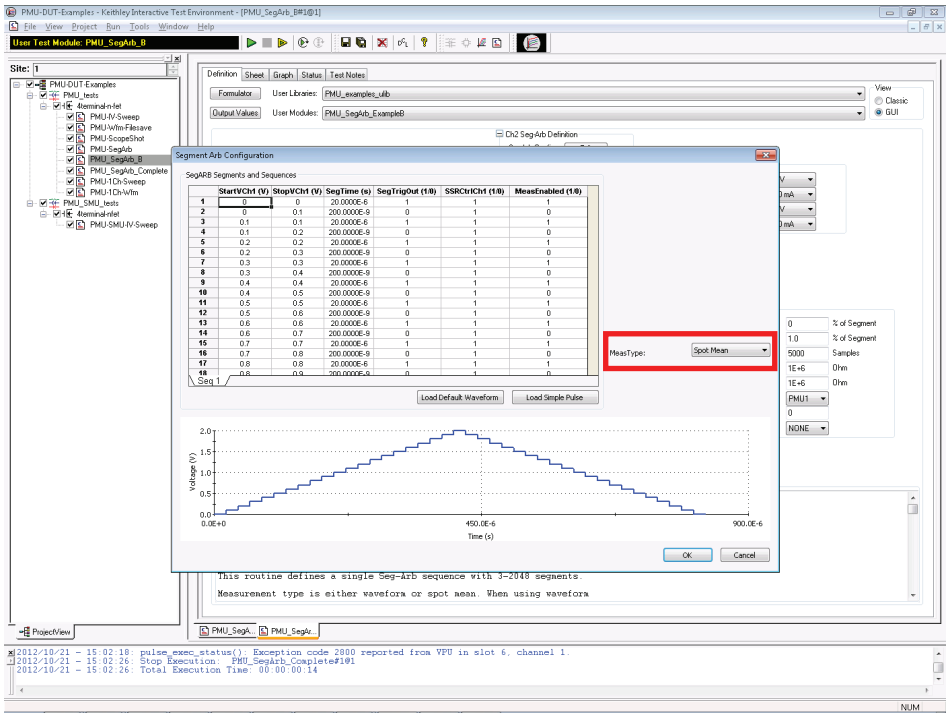


Figure 8-79
Two sequence Segment ARB waveform

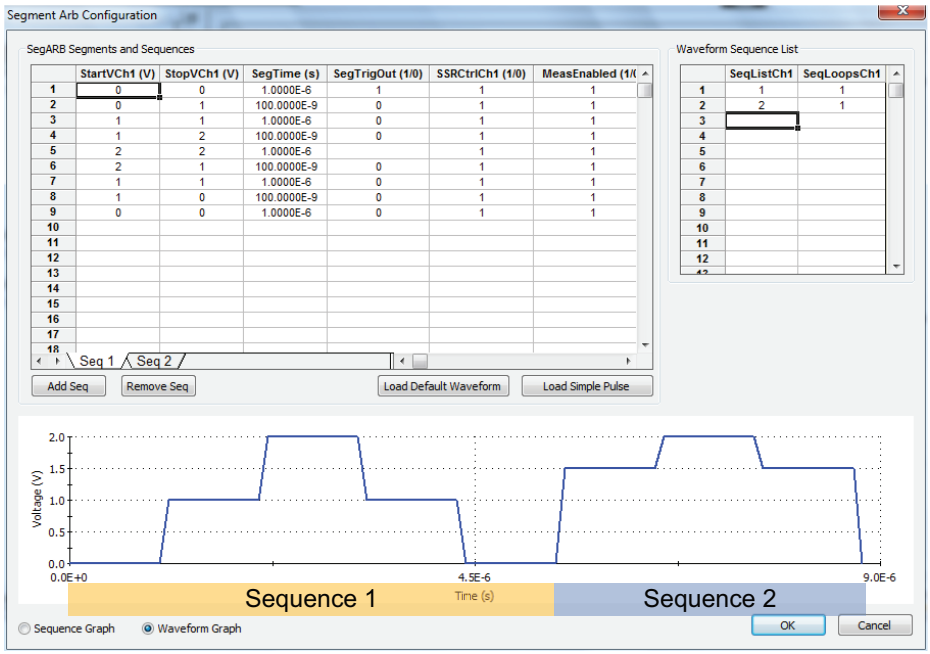
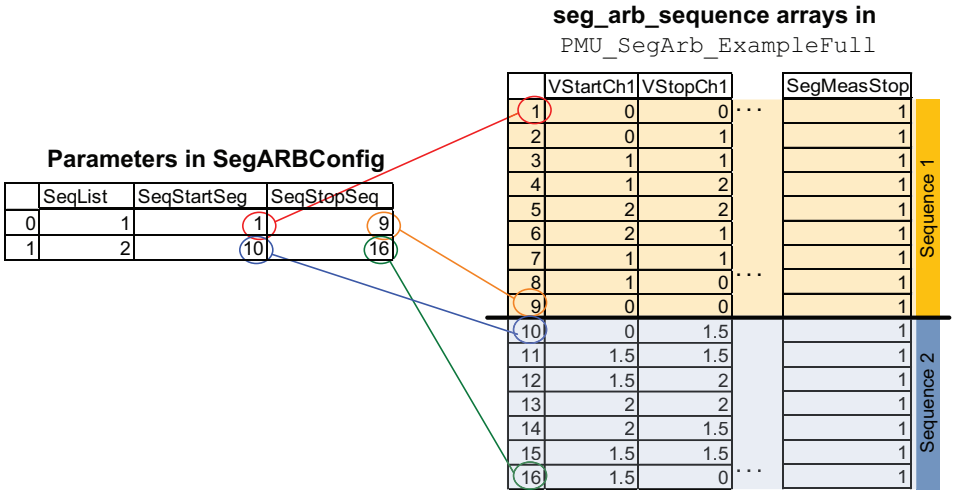


Figure 8-80
SegARBConfig uses SeqList, SeqStartSeg, and SeqStopSeq to store sequence



In this example for the user module PMU_SegArb_ExampleFull, several parameters are common across the two channels in the test:

- SegTime (SegTime)
- SegTrig (SegTrigOut)
- SegMeas (MeasEnabled)
- SegMeasType (MeasType)
- SegMeasStart (SegMeasStart)
- SegMeasStop (SegMeasStop)
- SeqList (SeqList)
- SeqStartSeg (SeqStartSeg)
- SeqStopSeg (SeqStopSeg)

Just like the other Control Types, to appear in the UTM GUI, the SegARBConfig must be assigned to a Group. If a user module has more than one Segment ARB channel, place a SegARBConfig control for each channel in a separate group. Choose which channel is configured by selecting the specific variable name. Figure 8-81 shows the SegARBConfig for user module channel one; Figure 8-82 shows the settings for channel two. For this example, selecting the variable named StartVCH1 configures the SegStartV for channel one, where selecting the variable named StartVCH2 configures the SegStartV for channel two. Make sure to select appropriate variables for each channel's SegARBConfig parameter assignment, otherwise unexpected test behavior may occur. Although the seg_arb_sequence command supports a maximum of 512 sequence definitions for each channel, the SegARBConfig control only supports 64 unique sequences for each channel. For the definition of the Segment ARB® waveform, the SegARBConfig control does support the full 512 sequences in the sequence list (SelSeqList) (see “seg_arb_waveform” on page 8-206).

Figure 8-81
Channel 1 SegARBConfig parameter configuration

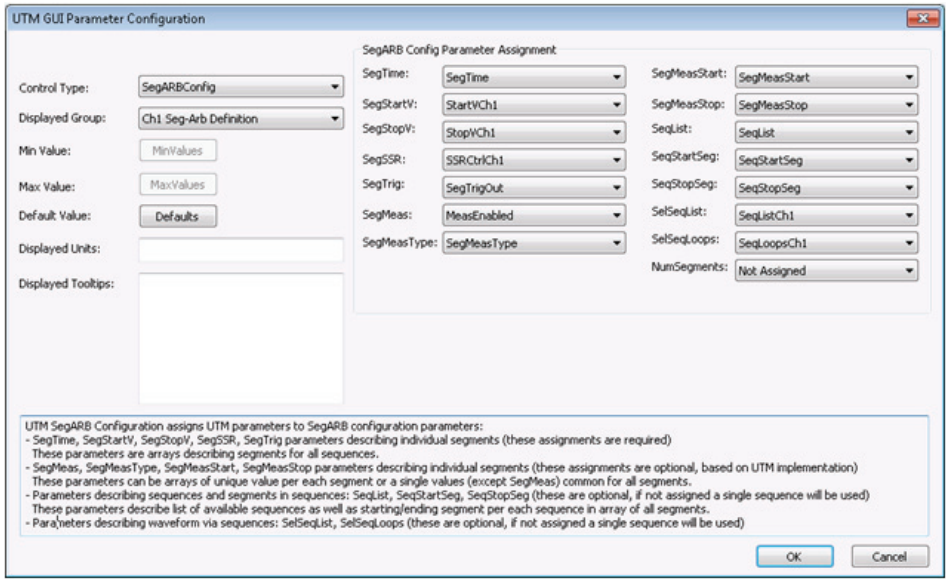
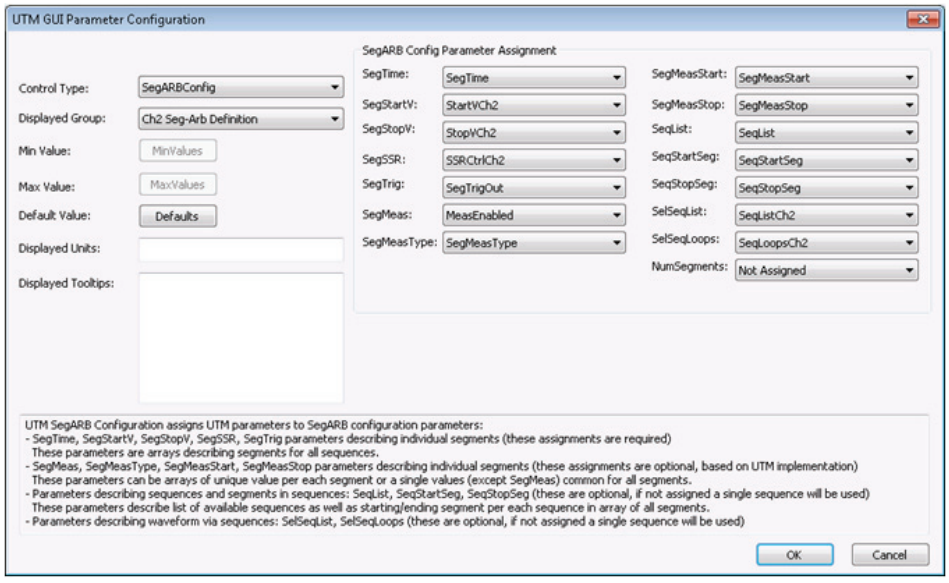


Figure 8-82
Channel 2 SegARBConfig parameter configuration



The SegARBConfig dialog can supply default waveforms (click the **Defaults** button shown in Figure 8-82). You can use the created defaults to get started with a new UTM without having to input any parameter values. Figure 8-83 shows the created blank Segment ARB Default dialog. To get a basic default waveform, click the **Load Simple Pulse** button (Figure 8-83). This results in the four segment pulse shown in Figure 8-84.

NOTE *The stop voltage of a segment must equal the start voltage of the next segment.*

Figure 8-83
Segment ARB Defaults Configuration

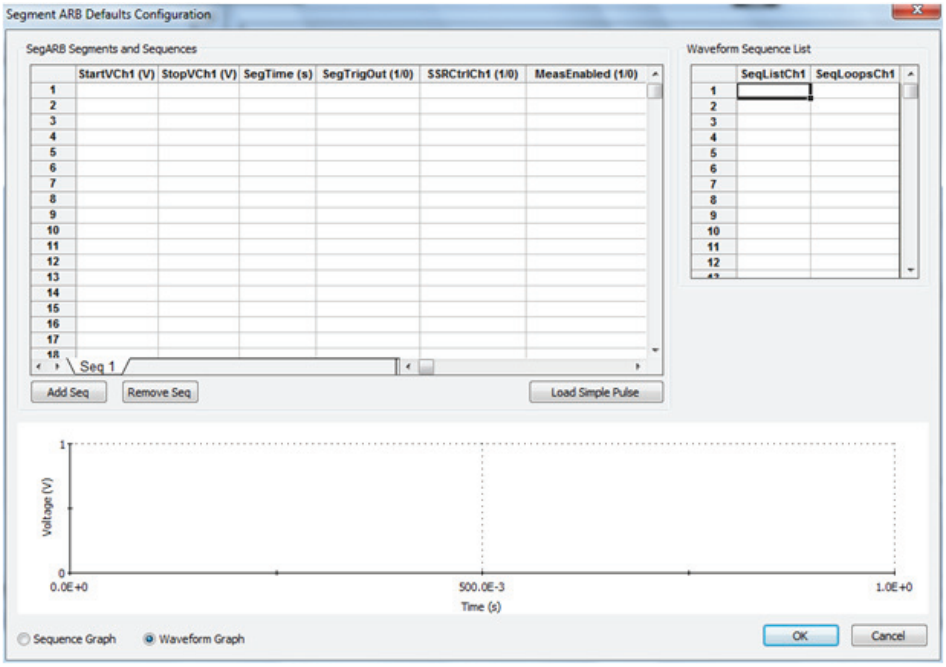


Figure 8-84
UTM GUI Editor Segment ARB Defaults Configuration (after pressing Load Simple Pulse

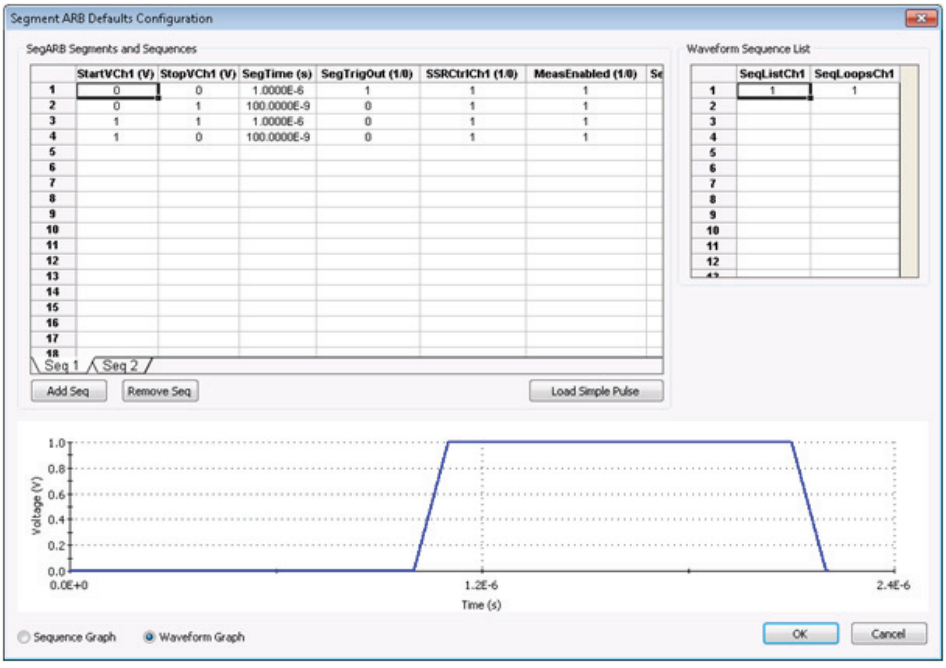


Figure 8-85 and Figure 8-86 show the defaults for this example user module PMU_SegArb_ExampleFull. When creating default waveforms, test them to make sure they properly run and provide the intended waveforms.

Figure 8-85
UTM GUI Editor Segment ARB Defaults Configuration for channel 1

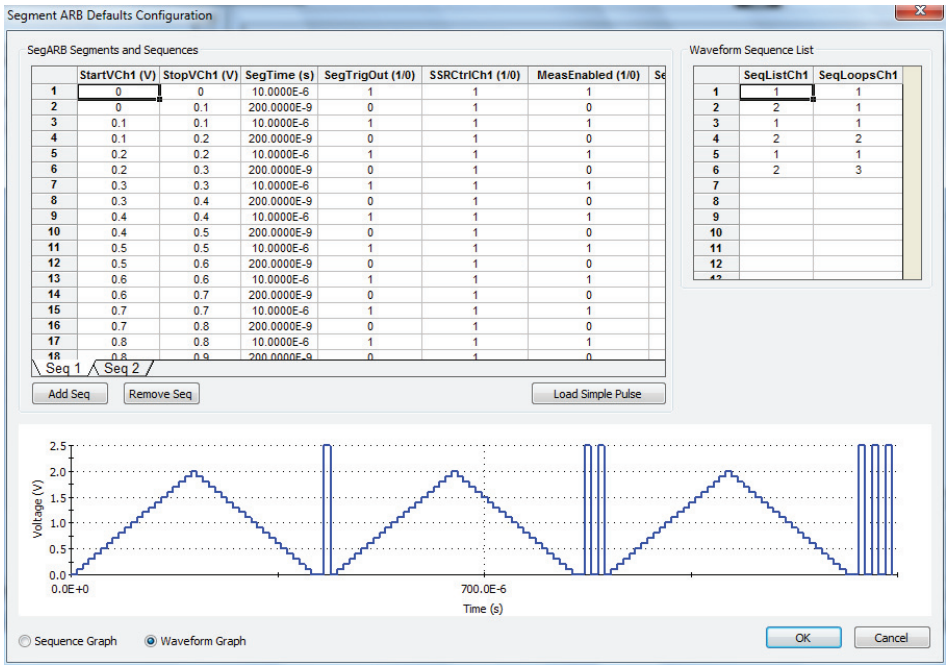
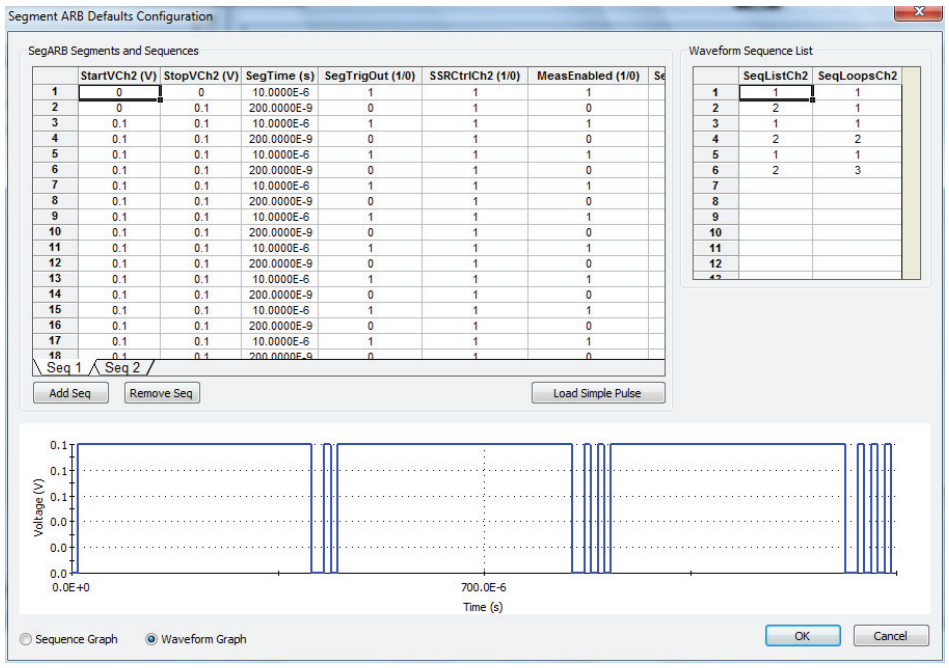
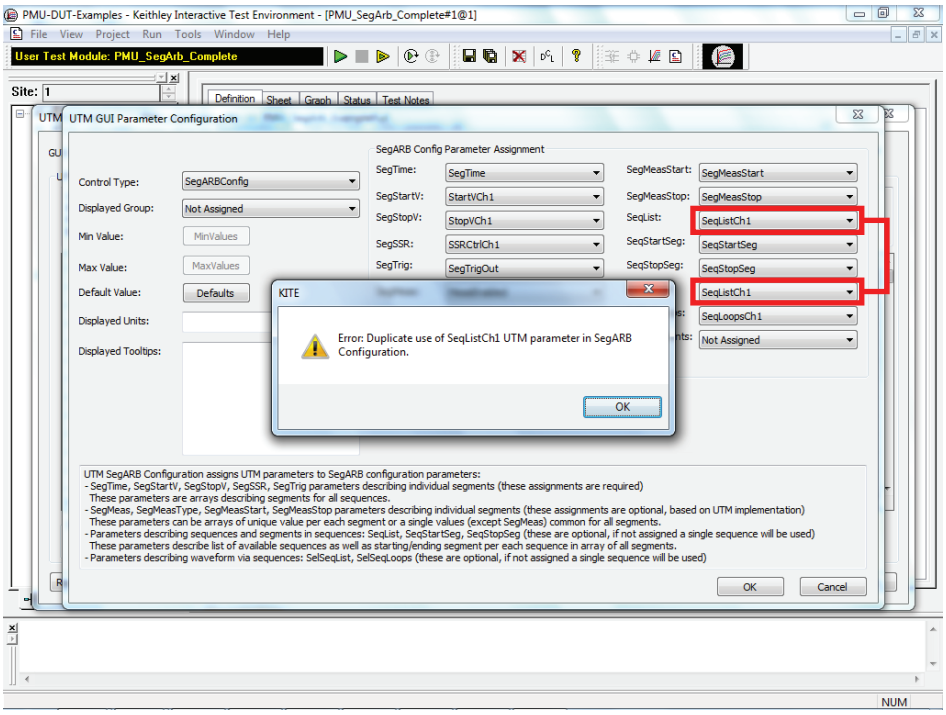


Figure 8-86
UTM GUI Editor Segment ARB Defaults Configuration for channel 2



Use care when configuring the SegARBConfig control by assigning the user module parameters to the SegARBConfig parameter names. Certain error checks are performed to minimize errors (correct any errors before the configuration can be accepted). Since it is not possible to catch all potential miss assignments of parameters, the correct configuration relies on the selection of correct parameters chosen (as shown in Figure 8-81 and Figure 8-82). For example, the configuration checks for duplicate parameters specified within a single channel. Figure 8-87 shows an error when using the parameter SeqListCh1 (in red boxes) twice in the Parameter Assignment.

Figure 8-87
Error as the result of duplicate use of parameter



UTM GUI definition file information

The UTM GUI Definition is stored as an XML file in the source directory of the user library. There are two possible files for the definition: Factory and user. The factory file has the filename user_module_name_GUI_Config.xml. For example, one factory UTM GUI file is PMU_examples_ulib_GUI_Config.xml. The user GUI filename format is: user_module_name_User_GUI_Config.xml. Both files are located in the src directory of the user module.

For example, for the above example PMU_examples_ulib, the XML files are stored in the directory C:\s4200\kiuser\usrlib\PMU_examples_ulib\src. The factory file stores UTM GUI definitions for all user modules in the user library. Modifying a UTM GUI for user module that has a factory UTM GUI will automatically create a User XML file. If a user definition exists for a user module, it will be used for the UTM GUI. For GUI definitions provided with the 4200-SCS, there will be a factory file. If a user module does not have a factory UTM GUI definition, creating a definition will create a user file. Do not modify the XML files outside of the UTM GUI Editor, as errors or non-functional UTM GUI definitions may result.

Reset defaults

On the main screen of the UTM GUI Editor, there is a Reset Defaults button (Figure 8-59). Click this button to use to overwrite the settings for the user module with the factory defaults from the factory UTM GUI definition file. If there is no factory definition for the user module, one is dynamically generated. Click the OK button on the UTM GUI Editor to save the changes made by pressing the Reset Defaults button.

Copying or moving UTM GUI definitions

The UTM GUI definition consists of one file for each user library. Use the kultcopy command to move a user library to another Model 4200. This command copies the user modules (c code), UTM GUI definition (XML files), and any bitmaps for the UTM GUI (bitmapped files with the .bmp extension). For more information, see kultcopy in "Copying user libraries using kultcopy" on page 8-48.

LPT Library Function Reference

The Keithley Instruments Linear Parametric Test Library (LPTLib) is a high-speed data acquisition and instrument control software library. It is the programmer's lowest level of command interface to the system's instrumentation. Its functions let the user configure the relay matrix and instrumentation to perform parametric tests.

This section lists the functions included in the Keithley Instruments LPTLib and describes how to use them. The descriptions contained here follow the general pattern:

- A purpose and format of each argument.
- Remarks in which detailed information about the function, along with the function's placement and relationship to other functions in a test sequence are described.
- Examples showing a typical use of the function in a test sequence.

Throughout this manual, the following conventions are used when explaining the functions:

- **All LPTLib functions are case-sensitive and must be entered as lower case when writing program codes.**
- **A capital letter X shown in a function name indicates** that the user must select from a list of replacement suffixes. Using the example `forceX`, the `X` can be replaced with either a `v` for voltage or `i` for current. The following is a table of possible suffixes, the parameter (function) each represents, and the units used throughout LPTLib for that parameter.

Table 8-10
Possible suffixes

Suffix	Parameter	Unit
`i`	Current	Amperes
`t`	Time	Seconds
`v`	Voltage	Volts

- **Brackets []** are used to enclose optional arguments of a function.
- **Period strings (...)** indicate additional arguments or functions that can be added.
- **Periods (.)** indicate data not shown in an example because it is not necessary to help explain the specific function.

NOTE *In this section, `10 point Courier italic` distinguishes the parameters in the Linear Parametric Test Library (LPTLib) function prototype from other elements.*

In this section, **`10 point Courier bold`** *highlights references to the name of the LPTLib function being described, as well as references to names of associated LPTLib functions.*

Using source compliance limits

When sourcing voltage (`forcev`), a current compliance limit can be set. When sourcing current (**`forcei`**), a voltage compliance limit can be set. The SMU will not exceed the compliance limits.

The **`limitX`** functions are used to set the compliance limits (**`limiti`** sets current compliance and **`limitv`** sets voltage compliance). The actual compliance limit that is in effect depends on the present measurement range. If the SMU is on a range that can accommodate the compliance limit value, then that compliance limit will be in effect. If the SMU is on a range that is too low for the compliance limit, the full scale value of the range will instead be used.

For example, if the compliance limit setting is 5 mA, but the SMU is on the 1mA range, the actual compliance will be 1.05 mA (full scale value). When the SMU upranges to the 10 mA range, the 5 mA compliance limit will go into effect.

There are two ways to perform a measure range change; manually select a range or use autorange. A range is manually selected by using a **`rangeX`** function. When a **`rangeX`** function is not used, the SMU will use autoranging when a measure function (**`intgX`**, **`measX`**, **`avgX`**, or **`bmeasX`**) is called.

The following procedures demonstrate the proper sequence for using the limit, range, source, and measure functions.

Autoranging: For autoranged measurements, perform the following steps to perform a source-measure operation:

1. Use a **`limitX`** function to set the desired compliance limit.
2. Use a **`forceX`** function to output a source value.
3. Use a measuring function (**`intgX`**, **`measX`**, or **`avgX`**) to perform an autoranged current or voltage measurement. Before performing the measurement, the SMU will first go to the most sensitive range for the measured reading.

For source-only functionality, a dummy measurement (Step 3 above) would have to be performed if a measurement uprange is required for a new compliance limit setting.

Manual ranging: When using a manual measurement range, perform the following steps to perform a source-measure operation:

1. Use a **`limitX`** function to set the desired compliance limit.
2. Use a **`rangeX`** function to select a manual measure range that will accommodate the compliance limit and the measured reading.
3. Use a **`forceX`** function to output a source value.
4. Use a measuring function (**`intgX`**, **`measX`** or **`avgX`**) to perform a current or voltage measurement.

For source-only operation, omit Step 4 of the above procedure.

LPT functions

In Table 8-11, function calls are grouped by function. The details about the functions for the SMUs and general operations are listed alphabetically after the table (see LPT functions for SMUs and general operations later in this section).

Next, details on functions for the pulse generators[7] are described:

- For the Model 4205-PG2, see LPT functions for the Model 4205-PG2.

7. The Models 4205-PG2, 4220-PGU, and 4225-PMU are a dual-channel pulse generator cards inside the Model 4200-SCS. To differentiate between an internal pulse card and other supported pulse instruments, a Model 4205-PG2 is referred to as a VPU (voltage pulse unit). With LPT functions, the Models 4205-PG2 and 4220-PGU are referred to as VPU1, VPU2, and so on. A Model 4225-PMU (pulse-measure unit) is referred to as a PMU (PMU1, PMU2, and so on).

- For the Models 4220-PGU and 4225-PMU, see LPT functions for the Models 4220-PGU and 4225-PMU.

NOTE *Table 15-44 provides the LPTLib function listing for the Model 4200-CVU. Details on the functions for the Model 4200-CVU follow the table.*

Table 8-11
Consolidated LPTLib function listing

Group	Function call
Instrument	devclr (Device clear) devint (Device initialize)
Matrix	addcon (Add connection) clrcon (Clear connection) conpin (Connect pin) conpth (Connect path) delcon (Delete connection)
Ranging	lorangeX (Define lowest range, `i,v`) rangeX (Range `i,v`) setauto (Re-enable autorange)
Sourcing	forceX (Force `i,v`) limitX (Limit `i,v`) mpulse (Generate pulse and measure output) pulseX (Generate pulse `i,v`)
Measuring	avgX (Average `i,v`) bmeasX (Block measure `i,v`) imeast (Immediate measure time) intgX (Integrate `i,v`) measX (Measure `i,t,v`)
Combination	asweepX (Array sweep `i,v`) bsweepX (Linear breakdown sweep `i,v`) clrscn (Clear scan) clrtrg (Clear trigger) rtfary (Return FORCE array) savgX (Sweep average `i,v`) scnmeas (Scan measure) searchX (Binary search `i,v`) sintgX (Sweep integrate `i,v`) smeasX (Measure `i,t,v`) sweepX (Linear sweep `i,v`) trigXg, trigXl (trigXg: trigger if `i,t,v` is ≥; trigXl: trigger if `i,t,v` is ≤)
Timing	adelay (Array delay) delay (Delay) disable (TIMER) (Time measurement function) enable (TIMER) (Time measurement function) rdelay (Real delay)
GPIB	kibcmd (Send GPIB command to instrument) kibdefclr (Clear instrument on `devclr`) kibdefdelete (Delete GPIB definition strings for `devclr` and `devint`) kibdefint (Clear instrument on `devint`) kibrcv (Read device dependent string) kibsnd (Send device dependent command) kibspl (Serial poll an instrument) kibsplw (Synchronous serial poll)
RS232	kspcfg (Configure the port) kspdefclr (Define string to clear RS-232 instrument on `devclr`) kspdefdelete (Delete RS-232 definition strings for `devclr` and `devint`) kspdefint (Define string to clear RS-232 instrument on `devint`) ksprcv (Send device dependent command string) kspsnd (Read device dependent string)

Table 8-11 (continued)
Consolidated LPTLib function listing

Group	Function call
General	getinstattr (Get configured instrument attributes) getinstid (Get instrument ID value from instrument name string) getinstname (Get instrument name string from instrument ID) GetKiteSite (Get KITE site number for site that is presently being tested) getstatus (Read system and instrument status information) setmode (Set operating mode) tstdsl (Test station de-select) tstsel (Test station select)
Execution*	execut (Execute) inshld (Instrument hold)
Arithmetic*	kfpabs (Floating point absolute value) kfpadd (Floating point add) kfpdiv (Floating point divide) kfpexp (Floating point raise e to a power) kfplog (Floating point return a natural logarithm) kfpmul (Floating point multiply) kfpneg (Floating point negative value) kfppwr (Floating point raise to a power) kfpsqrt (Floating point square root) kfpsub (Floating point subtract)
CVU (C-V measure)*	Table 15-44 provides the LPTLib function listing for C-V testing using the Model 4200-CVU. Details on using the CVU are covered in Section 15 of this manual.
PG2 (pulse only)*	Note: See LPT functions for the Model 4205-PG2 later in this section for details on the following functions: arb_array (Define a full-arb waveform) arb_file (Load a waveform from a full-arb waveform file) pg2_init (Reset PG2 to the specified pulse mode and its default settings) pulse_burst_count (Set burst mode pulse count) pulse_current_limit (Set current limit for pulse output) pulse_dc_output (Select DC output and set level) pulse_delay (Set time delay from trigger to pulse output) pulse_fall (Set pulse fall time) pulse_halt (Stops all pulse output) pulse_init (Reset Standard pulse to its default settings) pulse_load (Set output impedance of DUT) pulse_output (Set output channel on or off) pulse_output_mode (Set output mode to normal or complement) pulse_period (Set pulse period) pulse_range (Set voltage range for pulse low or pulse high) pulse_rise (Set pulse rise time) pulse_ssrc (Control the high endurance output relays for 4205-PG2) pulse_trig (Set trigger mode and initiate (or arm) pulse output) pulse_trig_output (Set trigger output on or off) pulse_trig_polarity (Set trigger output polarity) pulse_trig_source (Set trigger source) pulse_vhigh (Set pulse V High level) pulse_vlow (Set pulse V Low level) pulse_width (Set pulse width) seg_arb_define (Define a Segment ARB® waveform) seg_arb_file (Load a waveform from a Segment ARB waveform file)
Note: Triggering transition time must be <100 ns. Trigger output Level and Trigger In Level must be TTL.	

Table 8-11 (continued)
Consolidated LPTLib function listing

Group	Function call
PGU (pulse only) and PMU (pulse and measure)*	Note: See LPT functions for the Models 4220-PGU and 4225-PMU later in this section for details on the following functions: dev_abort (aborts a pulse_exec test) PostDataDouble (posts buffer data into KITE Sheet tab) PostDataDoubleBuffer (posts buffer data into KITE Sheet tab (large data sets)) pulse_chan_status (Returns the number of readings in the data buffer) pulse_conncomp (Controls connection compensation) pulse_exec (Starts test execution) pulse_exec_status (Checks the run status of the test) pulse_fetch (Retrieves enabled test data) pulse_limits (Sets voltage, current, and power thresholds) pulse_meas_sm (Configures spot mean measurements) pulse_meas_wfm (Configures waveform measurements) pulse_meas_timing (Sets the measurement windows) pulse_measrt (Returns data in pseudo real-time) pulse_ranges (Sets the pulse voltage and measure ranges) pulse_remove (Removes a channel from the test) pulse_sample_rate (Sets the sample rate) pulse_source_timing (Sets pulse source timing) pulse_step_linear (Configures pulse stepping type) pulse_sweep_linear (Configures pulse sweeping type) pulse_train (Configures a pulse train) rpm_config (Configures the Model 4225-RPM) seg_arb_sequence (Defines a Segment ARB pulse-measure sequence) seg_arb_waveform (Creates a Segment ARB pulse-measure waveform) setmode (Sets the number of iterations for LLEC (load line effect compensation))
* Provided for compatibility with other-platform versions of the LPT Library. The Models 4200-PG2 and 4205-PG2 are referred to as VPU1, VPU2, and so on in LPT functions. The Model 4220-PGU is referred to as VPU1, VPU2, and so on. The Model 4225-PMU is referred to as PMU1, PMU2, and so on.The Model 4200-CVU is referred to as CVU1, CVU2, and so on. Note that the Model 4225-PMU and Model 4225-PGU support the PG2 commands.	

LPT functions for SMUs and general operations

addcon Add connection

Purpose Add connections without clearing existing connections.

Format

```
int addcon(char exist_connect, int connect2, [connectn, [...]]
0);
```

`exist_connect` An instrument terminal ID. This instrument or terminal may have been, but is not required to have been, previously connected with `addcon`, `conpin`, or `conpth`.

`connect2` A pin number or an instrument terminal ID.

`connectn` A pin number or an instrument terminal ID.

Remarks `addcon` can be used to make additional connections on a matrix. `addcon` will simply connect every item in the argument list together, and there is no real distinction between `exist_connect` and the rest of the connection list. `addcon` behaves like the `conpin` command, except previous connections are never cleared.

Prior to making the new connections, `addcon` will clear all active sources by calling `devclr`.

The value -1 will be ignored by **addcon** and is considered a valid entry in the connection list; however, `exist_connect` may not be -1.

With the row-column connection scheme, only one instrument terminal may be connected to a pin.

See also `clrcon, conpin, conpth, delcon`

adelay Array delay

Purpose Specifies an array of delay points to use with `asweepX` calls. The delay is specified in units of seconds, with a resolution of 1 ms. The minimum delay is 0 s.

Format

```
int adelay(long delaypoints, double *delayarray);
```

`delaypoints` The number of separate delay points defined in the array.

`delayarray` The name of the array defining the delay points. This is a single dimension floating point array that is `delaypoints` long and contains the individual delay times. Units of the delays are seconds.

Remarks Each delay in the array is added to the delay specified in `asweep`. For example, if the array contained four delays (0.04 s, 0.05 s, 0.06 s, and 0.07 s) and the delay specified in `asweep` is 0.1 s, then the resulting delays are (0.14 s, 0.15 s, 0.16 s, and 0.17 s).

asweepX Array sweep

Purpose Generates a waveform based on a user-defined forcing array (logarithmic sweep or other custom forcing functions).

Format

```
int asweepi(int inst_id, long num_points, double delay_time,
double *force_array);

int asweepv(int inst_id, long num_points, double delay_time,
double *force_array);
```

`inst_id` The sourcing instrument's identification code.

`num_points` The number of separate current and voltage force points defined in the array.

`delay_time` The delay, in seconds, between each step and the measurements defined by the active measure list.

`force_array` The name of the user-defined force array. This is a single dimension array that contains all force points.

Remarks `asweepX` is used with `smeasX`, `sintgX`, or `savgX` functions.

`trigXl` and `trigXg` can also be used with `asweepX`. However, once a trigger point is reached, the sourcing device stops moving through the array. The output is held at the last forced point for the duration of the `asweep`. Data resulting from each step is stored in an array, as noted above, with `smeasX`. After the trigger point is reached,

measurements are made at each subsequent point. Results are approximately equal since the source is held at a constant output.

`asweepv` and `asweepi` are sourcing-type functions. When called, an automatic limit is imposed on the sourcing device. Refer to the `limit` command for additional information.

The maximum number of times data is measured (using `smeasX`, `sintgX`, or `savgX`) is determined by the `num_points` argument in `asweepX`. A one-dimensional result array with the same number of data elements as the selected value of `num_points` must be defined in the test program.

When multiple calls to `asweepX` are executed in the same test sequence, the `smeasX`, `sintgX`, or `savgX` arrays are loaded sequentially. This appends the measurements from the second `asweepX` to the previous results. If the arrays are not dimensioned correctly, access violations will occur. The measurement table remains intact until `devint`, `clrscn`, or `execut` are executed.

Defining new test sequences using `smeasX`, `sintgX`, or `savgX` appends the command to the active measure list. Previous measures are still defined and will be used. `clrscn` is used to eliminate previous buffers for the second sweep. Using `smeasX`, `sintgX`, and `savgX` after a `clrscn` call will cause the appropriate new measures to be defined and used.

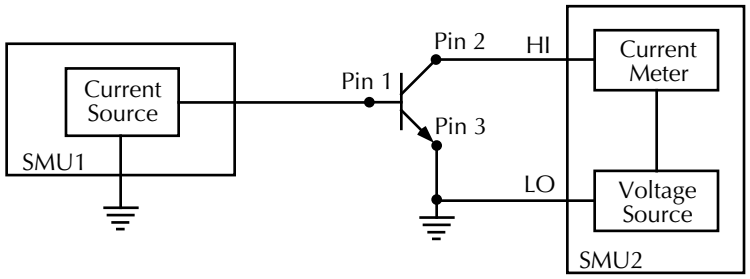
Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

If `adelay` is called prior to `asweepX`, each `adelay` value is added to the `asweepX` `delay_time`. This sum is compared to the maximum delay for the configured instrument card and if any value is larger, an error will occur. The SMU maximum delay is 2,147.483 s. The CVU maximum is 999 s.

Example

The following example gathers data to construct a graph showing the gain of a bipolar device over a wide range of base currents. A fixed collector-emitter bias is generated by SMU2. A logarithmic base current from 1.0E-10 to 1.0E-1A is generated by SMU1 using **asweepi**. The collector current applied by SMU2 is measured 10 times by **smeasi**. The data gathered is then stored in the ICMEAS array.

Figure 8-88
Diagram




```
double icmeas[10], ifrc[10];
.
.
ifrc[0]=1.0e-10;
for (i=1; i<10; i++)/* Create decade array from */
/* 1.0E-10 to 1.0E-1. */
    ifrc[i]=10.0*ifrc[i-1];
.
```

```
.
conpin(SMU1, 1, 0);/* Base connection. */
conpin(SMU2, 2, 0);/* Collector connection. */
conpin(GND, 3, 0);
limiti(SMU2, 200.0E-3);/* Reset I limit to maximum. */
smeasi(SMU2, icmeas);/* Define collector current */
/* array. */
forcev(SMU2, 5.0);/* Force vce bias. */
asweepi(SMU1, 10, 10.0E-3, ifrc);/* SweepIB, 10 points, 10 ms */
/* apart. */
execut();
```

avgX Average

Purpose

Performs a series of measurements and averages the results.

Format

```
int avgi(int inst_id, double *result, long stepno, double
steptime);
```

```
int avgv(int inst_id, double *result, long stepno, double
steptime);
```

`inst_id`	The instrument identification code of the measuring instrument.
`result`	The variable assigned the measurement's result.
`stepno`	The number of steps averaged in the measurement. This number ranges from 1 through 32,767.
`steptime`	The interval in seconds between each measurement. The minimum practical time is approximately 2.5 µs.

Remarks

`avgX` is used primarily to obtain measurements where:

- The DUT being tested acts in an unstable manner.
- Electrical interference is higher than can be tolerated if `measX` were to be used.

The programmer specifies the number of samplings and the duration between each sampling.

After this function executes, all closed relay matrix connections remain closed and the sources continue to generate voltage or current. This allows additional sequential measurements.

In general, measurement functions which return multiple results are more efficient than performing multiple measurement functions.

`rangeX` directly affects the operation of `avgX`. The use of `rangeX` prevents the instrument addressed from automatically changing ranges. This can result in an overrange condition such that would occur when measuring 10.0 V on a 4.0 V range. An overrange condition returns the value `1.0E+22` as the result of the measurement.

If `rangeX` is not located in the test sequence before the `avgX` call, the measurements performed automatically select the optimum range.

Compliance limits: A compliance limit setting goes into effect when the SMU is on a measure range that can accommodate the limit value. For manual ranging, the `rangeX` function is used to select the range. For autoranging, the `avgi` or `avgv` function will trigger a needed range change before the measurement is performed. See Using source compliance limits earlier in this section for details.

Example This example illustrates how the avgX command could be used to take five current readings and return the average of the measurements to the variable leakage:

```
double leakage;

.
.
limiti(SMU1, 1.0e-06);/* Limit the maximum current */
 /* to 1µA */
forcev(SMU1, 10.0);/* Force 10 V across the DUT */
delay(100);/* Delay 100 mS to allow for */
/* device settling */
avgi(SMU1, &leakage, 5, 0.01);/* Average 5 readings, delay */
/* 10 mS per measurement */
```

bmeasX Block measure

Purpose Takes a series of readings as fast as possible. This measurement mode allows for waveform capture and analysis (within the resolution of the measurement instrument).

Format

```
int bmeasi(int inst_id, double *results, long numrdg,
double delay, int timerid, double *timerdata);

int bmeasv(int inst_id, double *results, long numrdg,
double delay, int timerid, double *timerdata);
```

inst_id	The measuring instrument identification code.
results	The result name of array to receive readings. The array must be large enough to hold readings.
numrdg	The number of readings to return in the array.
delay	The delay between points to wait (in seconds).
timerid	The device name of the timer to use (0 = No timer data desired).
timerdata	The array used to receive the time points at which the readings were taken. If TimerID = 0, the timer is not read and this array is not updated. If used, the array must be large enough to hold readings.

Remarks This function collects data using the range presently selected. The measurement range is typically the same as the force range. If a different range is desired, you are required to place the instrument on the desired range prior to calling bmeasX.

When used with the Time Module, the measurements and the time when each measurement is performed are stored. The specific timer is defined in the function, and the time array is returned with the result array.

Example The example below shows how bmeas is used with a timer. Each measurement is associated with a time stamp. This time stamp marks the interval when each reading is made. This information is useful when determining how much time was required to obtain a specific reading.

```
double irange, volts, rdng[5], timer[5];
:
.
.
enable(TIMER1);/* Enable timer module. */
.
```

```
.
conpin(GND, 11, 0);/* Make connections. */
conpin(SMU3, 14, 0);
.
.
forcev(SMU3, volts);/* Perform test. */
measi(SMU3, &irange);/* Set I range of SMU based */
rangei(SMU3, irange);/* on initial measurement. */
.
forcev(SMU3, volts);
bmeasi(SMU3, rdng, 5, .0001,/* Block I measurement of 5 */
 TIMER1, timer);/* readings using SMU3 with */
/* 100µs delay between */
/* readings, using TIMER1 with */
/* time data labeled timer. */
```

Example Using no timer.

This example shows how the `bmeas` function is used without a timer. When used without a timer, the returned measurement is not associated with a time stamp.

```
double volts, rdng [5];
:
.
conpin(GND, 11, 0);/* Make connections. */
conpin(SMU3, 14, 0);
.
forcev(SMU3, volts);/* Perform test. */
.
bmeasi(SMU3, rdng, 5, 0, 0, 0);/* Block current measurement */
/* of 5 readings using SMU3. */
```

bsweepX Breakdown sweep

Purpose Supplies a series of ascending or descending voltages or currents and shuts down the source when a trigger condition is encountered.

Format

```
int bsweepi(int inst_id, double startval, double endval, long
num_points, double delay_time, double *result);
```

```
int bsweepv(int inst_id, double startval, double endval, long
num_points, double delay_time, double *result);
```

`inst_id`	The sourcing instrument's identification code.
`startval`	The initial voltage or current level applied as the first step in the sweep. This value can be positive or negative.
`endval`	The final voltage or current level applied as the last step in the sweep. This value can be positive or negative.
`num_points`	The number of separate current and voltage force points between `startval` and `endval`. The range may be from 1 through 32,767.
`delay_time`	The delay in seconds between each step and the measurements defined by the active measure list.
`result`	Assigned to the result of the trigger. This value represents the source value applied at the time of the trigger or breakdown.

Remarks

`bsweepX` is used with `trigXg`, `trigXl`, or `trigcomp`. These trigger functions are used to provide the termination point for the sweep. At the time of trigger or breakdown, all sources are shut down to prevent damage to the device under test. Typically, this termination point is the test current required for a given breakdown voltage.

Once triggered, `bsweepX` terminates the sweep and clears all sources by executing a `devclr` command internally. The standard `sweepX` command will continue to force the last value. This is useful for device characterization curves, but can cause problems when used in device breakdown conditions.

`bsweepX` can be used with `smeasX`, `sintgX`, `savgX`, or `rtfary` functions. Measurements are stored in a one-dimensional array in the consecutive order in which they were taken.

The system maintains a measurement scan table consisting of devices to test. This table is maintained using the `smeasX`, `sintgX`, `savgX`, or `clrscn` calls. As multiple calls to `sweep` functions are made, these commands are appended to the measurement scan table. Measurements are taken after the programmed `delay_time` is performed at the beginning of each `bsweepX` step.

When multiple calls to `bsweepX` are executed in the same test sequence, the arrays defined by `smeasX`, `sintgX`, or `savgX` calls are all loaded sequentially. The results from the second `bsweepX` call are appended to the results of the previous `bsweepX`. This can cause access violation errors if the arrays were not dimensioned for the absolute total. The measurement scan table remains intact until a `devint`, `clrscn`, or `execut` command completes.

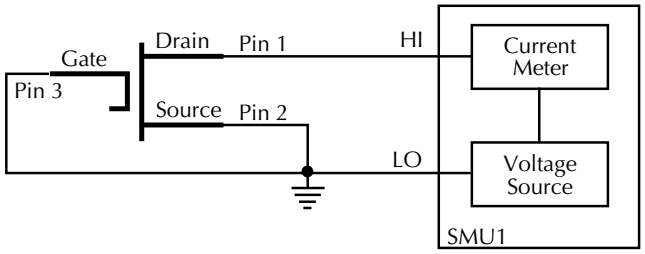
Defining new test sequences using `smeasX`, `sintgX`, or `savgX` adds the command to the active measure list. The previous measures are still defined and used; however, previous measures for the second sweep can be eliminated by calling `clrscn`. New measures are defined and used by calling `smeasX`, `sintgX`, or `savgX` after `clrscn`.

Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

Example

The following example measures the drain to source breakdown voltage of a FET. A linear voltage sweep is generated from 10.0 V to 50.0 V by SMU1 using the `bsweepv` function. The breakdown current is set to 10 mA by using `trigil`. The voltage at which this current is exceeded is stored in the variable `bvdss`.

Figure 8-89
bsweepv function example




```
double bvdss;
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 3, 0);
limiti(SMU1, 100.E-6);/* Define I limit for device. */
```

```
rangei(SMU1, 100.E-6);/* Select fixed range */
/* measurement. */
trigil(SMU1, -10.E-6);/* Set trigger point to 10 mA. */
bsweepv(SMU1, 10.0, 50.0, 40,/* Sweep from 10 to 50 V in 40 */
 10.0E-3, &bvdss);/* steps with 10 ms settling */
/* time per step. */
```

clrcon Clear connections

Purpose Opens or de-energizes all DUT pin and instrument matrix relays, disconnecting all crosspoint connections. If any sources are actively generating current or voltage, `devclr` is automatically called before the relay matrix is de-energized.

Format

```
int clrcon(void);
```

Remarks `clrcon` is called automatically by `devint` and `execut`. The first in a series of one or more connection type functions automatically calls a `clrcon`. Because this function is automatically called, it is not normally used by a programmer.

clrscn Clear scan table

Purpose Clears the measurement scan tables associated with a sweep.

Format

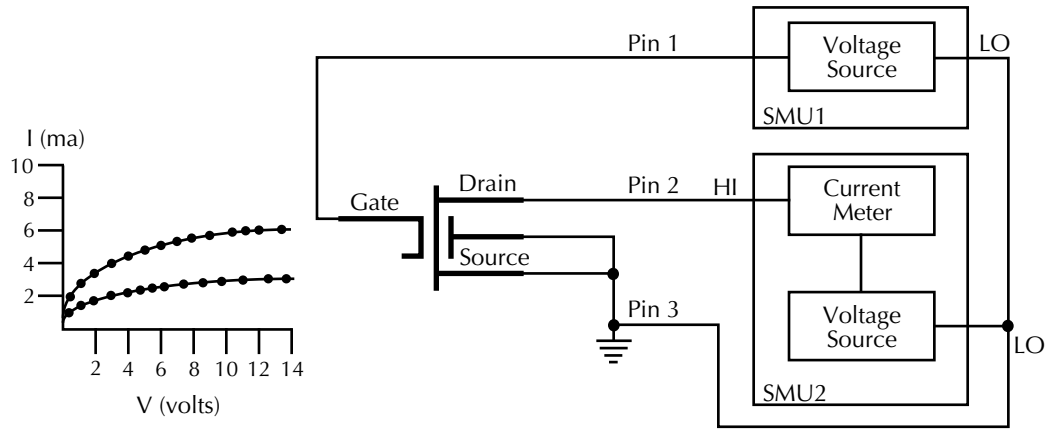
```
int clrscn(void);
```

Remarks When a single `sweepX` is used in a test sequence, there is no need to program a `clrscn` because `execut` clears the table. `clrscn` is only required when multiple sweeps and multiple sweep measurements are used within a single test sequence.

Example In the following example, `sweepX` configures SMU2 to source a voltage that sweeps from 0 through +14V in 14 steps. The results of the first `sweepv` are stored in an array called `res1`. Because of `clrscn`, the data and pointers associated with the first `sweepv` are cleared. Then 5 V are forced to the gate, and the measurement process is repeated. Results from these second measurements are stored in an array called `res2`.

This example obtains the measurement data needed to construct a graph showing an FET's gate voltage-to-drain current characteristics. The program samples the current generated by SMU2 14 times. This is done in two phases: First with 4 V applied to the gate, and second with 5 V applied. The gate voltages are generated by SMU1.

Figure 8-90
sweepX



```
double res1[14], res2[14];
.
conpin(SMU1, 1, 0);
conpin(SMU2, 2, 0);
conpin(GND, 3, 0);
forcev(SMU1, 4.0);/* Apply 4V to gate. */
smeasi(SMU2, res1);/* Measure drain current in */
/* each step; store results */
/* in res1 array. */
sweepv(SMU2, 0.0, 14.0, 13,
 2.0E-2);/* Perform 14 measurements */
/* over a range 0 through 14V. */
clrscn();/* Clear smeasi. */
forcev(SMU1, 5.0);/* Apply 5 V to gate. */
smeasi(SMU2, res2);/* Measure drain current in */
/* each step; store results in */
/* res2 array. */
sweepv(SMU2, 0.0, 14.0, 13,
 2.0E-2);/* Perform 14 measurements */
/* over a range 0 through 14V. */
```

clrtrg Clear trigger

Purpose Clears the user-selected voltage or current level used to set trigger points. This permits the use of `trigXl` or `trigXg` more than once with different levels within a single test sequence.

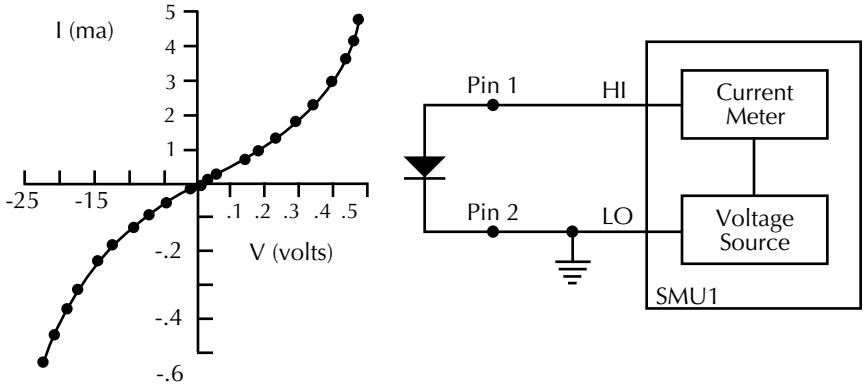
Format

```
int clrtrg(void);
```

Remarks `searchX`, `sweepX`, `asweepX`, or `bsweepX`, each with different voltage or current levels, may be used repeatedly within a function provided each is separated by a `clrtrg`.

Example The following example collects data and constructs a graph showing the forward and reverse conduction characteristics of a diode. `clrtrg` allows multiple triggers to be programmed twice in the same test sequence. Each result is returned to a separate array.

Figure 8-91
clrtrg example



```
double forcur [11], revcur [11];/* Defines arrays. */
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 0);
trigil(SMU1, 5.0e-3);/* Increase ramp to I = 5 mA.*/
smeasi(SMU1, forcur);/* Measure forward */
/* characteristics; */
/* return results to forcur */
/* array. */
sweepv(SMU1, 0.0, 0.5, 10,/* Output 0 to 0.5 V in 10 */
 5.0e-3);/* steps, each 5 ms duration. */
clrtrg();/* Clear 5 mA trigger point. */
clrscn();/* Clear sweepv. */
trigil(SMU1, -0.5e-3);/* Decrease ramp to */
/* I = -0.5 mA. */
smeasi(SMU1, revcur); /* Measure reverse */
/* characteristics; */
/* return results to revcur */
/* array. */
sweepv(SMU1, 0.0, -30.0, 10,   /* Output 0 to -30 V in 10 steps
*/
 5.00e-3); /* each 5 ms in duration. */
```

conpin Connect pin

Purpose Connect pins and instruments together.

Format

```
int conpin(char InstrTermID, int connect1, int connect2,
[connectn, [...]] 0);
```

`InstTermID` The instrument terminal ID. For example: SMU1, GNDU, or, PMU1CH1.

`connect1` A pin number or an instrument terminal ID.

`connect2` A pin number or an instrument terminal ID.

`connectn` A pin number or an instrument terminal ID.

Remarks `conpin` will simply connect every item in the argument list together. As long as there are no connection rules violated, the pin or terminal will be connected to the additional items along with everything to which it is already connected.

The first `conpin` or `conpth` after any other LPT call will clear all sources by calling `devclr` and then clear all matrix connections by calling `clrcon` before making the new connections.

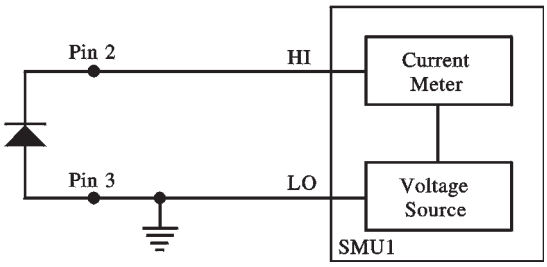
The value -1 will be ignored by `conpin` and is considered a valid entry in the connection list.

With the row-column connection scheme, only one instrument terminal may be connected to a pin.

See also `addcon, clrcon, conpth, delcon`

Example

Figure 8-92
conpin example




```
conpin(3, GND, 0);/* Connect pin 3 to SMU1 */
/* and ground. */
conpin(2, SMU1, 0);/* Connect pin 2 to SMU1. */
.
.
```

conpth Connect path

Purpose Connect pins and instruments together using a specific pathway.

Format

```
int conpth(int path, int connect1, int connect2, [connectn,
[...]] 0);
```

`path`	Pathway number to use for the connections.
`connect1`	A pin number or an instrument terminal ID.
`connect2`	A pin number or an instrument terminal ID.
`connectn`	A pin number or an instrument terminal ID.

Remarks The system can be forced to use a particular pathway by using `conpth` instead of `conpin`. This might be done to provide additional electrical isolation between two connections. The eight pathways are numbered 1 through 8.

The first `conpin` or `conpth` after any other LPT call will clear all sources by calling `devclr` and then clear all matrix connections by calling `clrcon` before making the new connections.

The value -1 for any item in the connection list will be ignored by `conpth` and is considered a valid entry in the connection list.

The `conpth` function is not valid in the row-column connection scheme. When the matrix is configured for remote sense, the only valid path values are 1, 3, 5, and 7.

See also `addcon, clrcon, conpin, delcon`

delay Delay

Purpose Provides a user-programmable delay within a test sequence.

Format

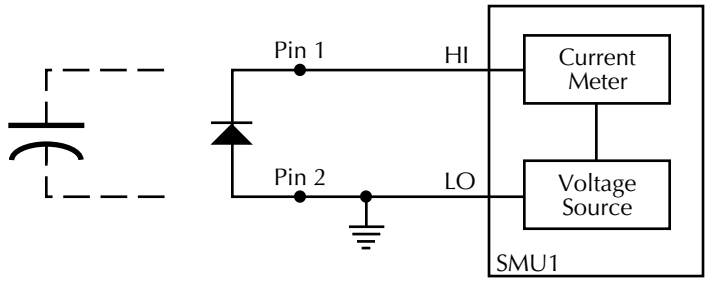
```
int delay(long n);
```

n	is the desired duration of the delay in milliseconds.

Remarks `delay` can be called anywhere within the test sequence.

Example This example measures a variable capacitance diode's leakage current. SMU1 applies 60 V across the diode. This device is always configured in the reverse bias mode, so the high side of SMU1 is connected to the cathode. This type of diode has very high capacitance and low leakage current; therefore, a 20 ms delay is added. After the delay, current through SMU1 is measured and stored in the variable IR4.

Figure 8-93
delay example




```
double ir4;
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 0);
forcev(SMU1, 60.0);/* Generate 60 V from SMU1. */
delay(20);/* Pause for 20 ms. */
measi(SMU1, &ir4);/* Measure current; return */
/* result to ir4. */
```

delcon Delete connection

Purpose Remove specific matrix connections.

Format

```
int delcon(char InstrTermID, int exist_connect, [int
exist_connectn, [...]] 0);
```

`InstTermID`	The instrument terminal ID. For example: SMU1, GNDU, or, PMU1CH1.
`exist_connect`	A pin number or an instrument terminal ID.
`exist_connectn`	A pin number or an instrument terminal ID.

Remarks All connections to each terminal or pin listed will be disconnected. Prior to making the disconnections, `delcon` will clear all active sources by calling `devclr`.

If GND is included in the list, all ground connections will be removed. If an SMU remains connected, GND must be reconnected using `addcon` or an error will be generated when the first LPTLib function after the connection sequence executes.

A programmer can perform a series of tests within a single test sequence using `addcon` and `delcon` together without breaking existing connections. Only the required terminal and pin changes are made before performing the next sourcing and measuring operations.

See also `addcon, clrcon, conpin, conpth`

Example

```
double i1, i2;

conpin(3, GND, 0);
conpin(1, SMU1, 0);
conpin(2, SMU2, 0);
forcev(SMU1, 1.0);
forcei(SMU2, .001);
measi(SMU1, &i1);
delcon(SMU2, 0);/* Remove SMU2 from the circuit */
forcev(SMU1, 1.0);/* because delcon cleared sources. */
measi(SMU1, &i2);
```

devclr Device clear

Purpose Sets all sources to a zero state.

Format

```
int devclr(void);
```

Remarks This function will clear all sources sequentially in the reverse order from which they were originally forced. Prior to clearing all Keithley supported instruments, GPIB based instruments will be cleared by sending all strings defined with `kibdefclr`.

`devclr` is implicitly called by `clrcon`, `devint`, `execut`, and `tstdsl`.

devint Device initialize

Purpose Resets all active instruments; clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to the default states. Refer to the specific hardware manuals for listings of available ranges and the default conditions and ranges for the instrumentation. To abort a running `pulse_exec` pulse test, see dev_abort.

Format

```
int devint(void);
```

Remarks This function will reset all active instruments in the system to their default states.

This function performs the following actions:

1) Clear all sources by calling `devclr`.
2) Clear the matrix crosspoints by calling `clrcon`.
3) Clear the trigger tables by calling `clrtrg`.
4) Clear the sweep tables by calling `clrscn`.
5) Reset GPIB instruments by sending the string defined with `kibdefint`.
6) Resets the active instrument cards in the following order
   a) SMU instrument cards
   b) CVU instrument cards
   c) Pulse instrument cards (Model 4225-PMU, Model 4220-PGU, and Model 4205-PG2)

Note that the pulse mode is maintained. For example, if the pulse card is in Segment ARB® mode, it will still be in Segment Arb mode after the devint process is complete.

Also, `devint` is implicitly called by `execut` and `tstdsl`; `devclr` is implicitly called by `clrcon`.

disable Disable timer

Purpose Stops the timer and sets the time value to zero (0). Timer reading is also stopped.

Format

```
intm disable(int inst_id);
```

`inst_id` The instrument ID of timer module (`TIMERn`).

Remarks `disable` (`TIMERn`) stops the timer and resets the time value to zero (0).

enable Initialize and start timer

Purpose Provides correlation of real time to measurements of voltage, current, conductance, and capacitance.

Format

```
int enable(int instr_id);
```

`instr_id` The instrument ID of timer module (`TIMERn`).

Remarks `enable` (`TIMERn`) initializes and starts the timer and allows other measurements to read the timer. The time starts at zero (0) at the time of the enable call.

execut Execute

Purpose Causes system to wait for the preceding test sequence to be executed.

Format

```
int execut(void);
```

Remarks This function will wait for all previous LPT commands to complete and then will issue a `devint`.

forceX Force a voltage or current

Purpose Programs a sourcing instrument to generate a voltage or current at a specific level.

Format

```
int forcei(int inst_id, double value);
int forcev(int inst_id, double value);
```

`inst_id` The instrument identification code. Refer to the instrument documents for the code.

`value` The level of the bipolar voltage or current forced in volts or amperes.

Remarks `forcev` and `forcei` generate either a positive or negative voltage, as directed by the sign of the value argument. With both `forcev` and `forcei`:

- Positive values generate positive voltage or current from the high terminal of the source relative to the low terminal.
- Negative values generate negative voltage or current from the high terminal of the source relative to the low terminal.

When using `limitX`, `rangeX`, and `forceX` on the same source at the same time in a test sequence, call `limitX` and `rangeX` before `forceX`. See Using source compliance limits for details.

The ranges of currents and voltages available from a voltage or current source vary with the specific instrument type. For more detailed information, refer to the specific hardware manual for each instrument.

To force zero current with a higher voltage limit than the 20 V default, include one of the following calls ahead of the `forcei` call:

- A `measv` call, which causes the Model 4200-SCS to autorange to a higher voltage limit.
- A `rangev` call to an appropriate fixed voltage, which results in a fixed voltage limit.

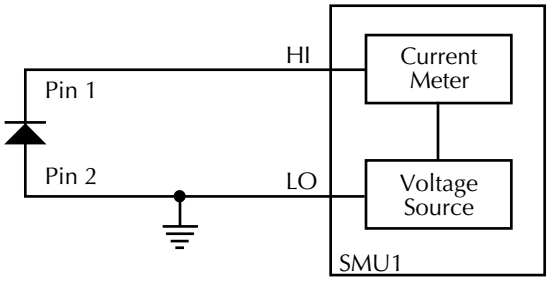
To force zero volts with a higher current limit than the 10 mA default, include one of the following calls ahead of the `forcev` call:

- A `measi` call, which causes the Model 4200-SCS to autorange to a higher current limit.
- A `rangei` call to an appropriate fixed current, which results in a fixed current limit.

Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

Example The reverse bias leakage of a diode is measured after applying 40.0 V to the junction.

Figure 8-94
forcei example




```
double ir12;
.
.
conpin(2, GND, 0);
conpin(SMU1, 1, 0);
limiti(SMU1, 2.0E-4);/* Limit 1 to 200µA. */
forcev(SMU1, 40.0);/* Apply 40.0 V. */
measi(SMU1, &ir12);/* Measure leakage; */
/* return results to ir12. */
```

getinstattr Get configured instrument attributes

Purpose All instruments in the system configuration have specific attributes. GPIB address is an example of an attribute. The values of these attributes change as the system configuration is changed. Therefore, by getting the values of key attributes at run time, user modules can be developed in a configuration-independent manner. Given an instrument identification code and an attribute name string, this module returns the specified attribute value string.

Format

```
int getinstattr(int inst_id, char *attrstr, char *attrvalstr);
```

`inst_id` The LPTLib instrument identifier (ID).

`attrstr` The instrument attribute name string.

`attrvalstr` The value string of the requested attribute. If the requested attribute exists, the returned string will match one of the values shown in the Attribute value string column of Table 8-8. If the requested attribute does not exist, the `attrvalstr` parameter will set to a null string.

Possible values for the `getinstattr` parameters are listed in Table 8-12.

Table 8-12
getinstattr parameter values

Instrument identification code	Attribute name string	Attribute value string
GPIx	GPIBADDR	1-30
	MODELNUM	GPI 2-Terminal GPI 4-Terminal
CMTRx	GPIBADDR	1-30
	MODELNUM	HP4980 KI590
PGUx	GPIBADDR	1-30
	MODELNUM	HP8110
SMUx	MODELNUM	KI4200 KI4210
MTRX1	MODELNUM	KI707 KI708
TF1	MODELNUM	KI8006 KI8007
	NUMOFPINS	12 72
PRBR1	NUMOFPINS	2-72
	MODELNUM	FAKE PA200 MANL MM40
CVUx	MODELNUM	KICVU KICVU4210
VPUx VPUxCH1 VPUxCH2	MODELNUM	KIVPU KIVPU4205 KIVPU4220
PMUx PMUxCH1 PMUxCH2	MODELNUM	KIPMU4225

getinstid Get instrument ID

Purpose Get the instrument identifier (ID) from the instrument name string.

Format

```
int getinstid(char *inst_name, int *inst_id);
```

inst_name The instrument name string.

inst_id The returned instrument identifier.

getinstname Get instrument name

Purpose Get the instrument name string from the instrument identifier (ID).

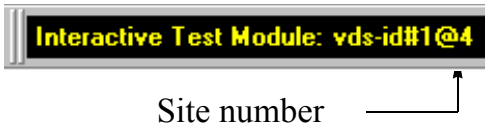
Format
```
int getinstname(int *inst_id, char *inst_name);
```
inst_id The instrument identifier.

inst_name The returned instrument name string.

GetKiteSite Get KITE site number

Purpose Get the site number for the site that KITE is presently testing: The number that KITE displays after the @ symbol in the Test/Plan Indicator box (above the Project Navigator). See Figure 8-95.

Figure 8-95
Getting the site number currently under test



Format
```
int GetKiteSite(void)
```

Remarks The `GetKiteSite` function returns the KITE site number; an integer that designates the relative numerical position of the presently tested site in the prober site-visit sequence. However, users normally correlate KITE site numbers with prober site coordinates; note that `GetKiteSite` does *not* return prober site coordinates.

For more information about KITE site numbers, refer to Section 6, Multi-site Project Plan execution and 'Run' execution of Project Plans.

getlpterr Get LPT error

Purpose Get the first LPT error since the last **devint**.

Format
```
int getlpterr(void);
```

Remarks This function will fetch the error code of the first error encountered since the last **devint** call.

Returns the first error code encountered since the last **devint**.

getstatus Get instrument status

Purpose Returns the operating state of the desired instrument.

Format
```
int getstatus(int inst_id, long parameter, double *result);
```
inst_id The instrument identifier (ID), such as SMU1, SMU2, VPU1, VPU2.

parameter The parameter of query.

result The data returned from the instrument. `getstatus` returns one item.

Remarks Valid errors:

The UT_INVLDPRM invalid parameter error is returned from getstatus. The status item parameter is illegal for this device. The requested status code is invalid for selected device.

A list of supported getstatus query parameters for an SMU and a pulse generator card (VPU) are provided in Table 8-13 and Table 8-14.

Table 8-13
Supported SMU getstatus query parameters

SMU parameter	Returns	Comment
KI_IPVALUE	The presently programmed output value	Current value (I output value)
KI_VPVALUE		Voltage value (V output value)
KI_IPRANGE	The presently programmed range	Current range (full-scale range value, or 0.0 for autorange)
KI_VPRANGE		Voltage range (full-scale range value, or 0.0 for autorange)
KI_IARANGE	The presently active range	Current range (full-scale range value)
KI_VARANGE		Voltage range (full-scale range value)
KI_COMPLNC	Active compliance status	Bitmapped values: 2 = LIMIT (at the compliance limit set by **limitX**). 4 = RANGE (at the top of the range set by **rangeX**).
KI_RANGE_COMPLIANCE	Active compliance status for fixed range	In range compliance if 1.
KI_COMPLNC_EVER	Compliance history	Reset by reading compliance history and by **devint**.

Table 8-14
Supported pulse generator card getstatus query parameters

Parameter		Comment
General parameters:		
KI_VPU_PERIOD	Pulse period	Pulse period value in seconds
KI_VPU_TRIG_POLARITY	Trigger polarity	Rising or falling edge
KI_VPU_CARD_STATUS		Card level status
KI_VPU_TRIG_SOURCE	Trigger source	Trigger source value
Channel based parameters:		
KI_VPU_CH1_RANGE	Source range	Channel 1 range value in volts (5.0 or 20.0)
KI_VPU_CH2_RANGE	Source range	Channel 2 range value in volts (5.0 or 20.0)
KI_VPU_CH1_RISE	Rise time	Channel 1 rise time value in seconds
KI_VPU_ CH2_RISE	Rise time	Channel 2 rise time value in seconds
KI_VPU_CH1_FALL	Fall time	Channel 1 fall time value in seconds
KI_VPU_ CH2_FALL	Fall time	Channel 2 fall time value in seconds
KI_VPU_CH1_WIDTH	Pulse width	Channel 1 pulse width value in seconds
KI_VPU_ CH2_WIDTH	Pulse width	Channel 2 pulse width value in seconds
KI_VPU_CH1_VHIGH	Pulse high	Channel 1 pulse high level value in volts
KI_VPU_ CH2_ VHIGH	Pulse high	Channel 2 pulse high level value in volts
KI_VPU_CH1_VLOW	Pulse low	Channel 1 pulse low level value in volts
KI_VPU_ CH2_ VLOW	Pulse low	Channel 2 pulse low level value in volts
KI_VPU_CH1_DELAY	Pulse delay	Channel 1 pulse delay from trigger value in seconds
KI_VPU_ CH2_DELAY	Pulse delay	Channel 2 pulse delay from trigger value in seconds

Table 8-14 (continued)
Supported pulse generator card getstatus query parameters

Parameter		Comment
KI_VPU_CH1_ILIMIT	Current limit	Channel 1 current Limit value in amps
KI_VPU_ CH2_ILIMIT	Current limit	Channel 2 current Limit value in amps
KI_VPU_CH1_BURST_COUNT	Burst count	Channel 1 burst count value
KI_VPU_ CH2_BURST_COUNT	Burst count	Channel 2 burst count value
KI_VPU_CH1_TEST_STATUS		Channel 1 test status
KI_VPU_CH2_TEST_STATUS		Channel 2 test status
KI_VPU_ CH1_DC_OUTPUT	DC output	Channel 1 DC output value
KI_VPU_ CH2_DC_OUTPUT	DC output	Channel 2 DC output value
KI_VPU_CH1_LOAD	Pulse load	Channel 1 pulse load value
KI_VPU_CH2_LOAD	Pulse load	Channel 2 pulse load value

imeast Immediate measure

Purpose Force a read of the timer and return the result.

Format
```
int imeast(int inst_id, double *result);
```
inst_id The device ID.

result The variable assigned to the measurement.

Remarks This command applies to all timers.

inshld Instrument hold

Purpose Provided for compatibility with Model S400 LPTlib.

Format
```
int inshld(void);
```

intgX Integrate

Purpose Performs voltage or current measurements averaged over a user-defined period (usually, one AC line cycle). This averaging is done in hardware by integration of the analog measurement signal over a specified period of time. The integration is automatically corrected for 50 Hz or 60 Hz power mains.

Format
```
int intgi(int inst_id, double *result);
int intgv(int inst_id, double *result);
```
inst_id The measuring instrument identifier (ID), for example, SMU1.

result The variable assigned to the result of the measurement.

Remarks For a measurement conversion, the signal is sampled for a specific period of time. This sampling time for measurement is called the integration time. For the `intgX` function, the default integration time is set to 1 PLC. For 60Hz line power, 1 PLC = 16.67 ms (1 PLC/60 Hz). For 50 Hz line power, 1 PLC = 20 ms (1 PLC/50 Hz).

The default integration time is one AC line cycle (1 PLC). This default time can be overridden with the `KI_INTGPLC` option of `setmode`. The integration time can be set

from 0.01 PLC to 10.0 PLC. The `devint` command resets the integration time to the one AC line cycle default value (1 PLC).

NOTE *The only difference between* `measX` *and* `intgX` *is the integration time. For* `measX`*, the integration time is fixed at 0.01PLC. For* `intgX`*, the default integration time is 1 PLC, but can set to any PLC value between 0.01 and 10.0.*

`rangeX` directly affects the operation of `intgX`. The use of `rangeX` prevents the instrument addressed from automatically changing ranges. This can result in an overrange condition that would occur when measuring 10.0 V on a 4.0 V range. An overrange condition returns the value 1.0E+22 as the measurement result.

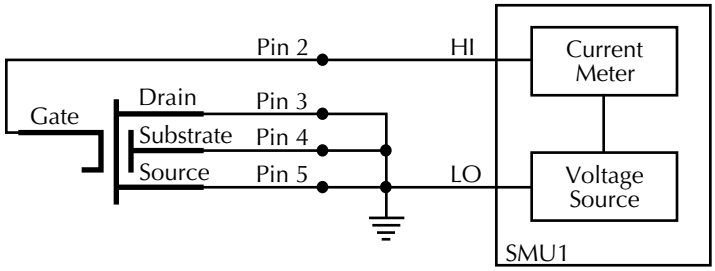
If used, `rangeX` must be located in the test sequence before the associated `intgX` function.

In general, measurement functions which return multiple results are more efficient than performing multiple measurement functions.

Compliance limits – A compliance limit setting goes into effect when the SMU is on a measure range that can accommodate the limit value. For manual ranging, the `rangeX` function is used to select the range. For autoranging, the `intgi` or `intgv` function will trigger a needed range change before the measurement is performed. See Using source compliance limits earlier in this section for details.

Example This example measures the relatively low leakage current of a MOSFET.

Figure 8-96
Measuring low leakage current of a MOSFET




```
double idss;
.
.
conpin(GND, 5, 4, 3, 0);
conpin(SMU1, 2, 0);
limiti(SMU1, 2.0E-8);/* Limits to 20.0nA. */
rangei(SMU1, 2.0E-8);/* Select range for 20.0nA */
forcev(SMU1, 25.0);/* Apply 25 V to the gate. */
intgi(SMU1, &idss);/* Measure gate leakage; */
/* return results to idss. */
```

kfpabs Floating point absolute value

Purpose Takes a user-specified positive or negative value and converts it into a positive value that is returned to a specified variable.

Format

```
int kfpabs(double *x, double *z);
```

`x` Pointer to the variable to be converted to an absolute value.

z	Pointer to the variable where the result will be stored.

Example This example takes the absolute value of a current reading. forcev outputs vb1 volts from SMU1. This current is measured with measi, and the result is stored in location ares2. The absolute value of ares2 is then calculated and stored as ares2.

```
double ares2, vb1;
.
.
forcev(SMU1, vb1);/* Output vb1 from SMU1. */
measi(SMU1, &ares2);/* Measure SMU1 current; */
/* store in ares2. */
kfpabs(&ares2, &ares2);/* Convert ares2 to absolute */
/* value; return result to ares2*/
```

kfpadd Keithley floating point add

Purpose Adds two real numbers and stores the result in a specified variable.

Format

```
int kfpadd(double *x, double *y, double *z);
```

x	The first of two values to add.
y	The second of two values to add.
z	A variable in which the sum x + y will be stored.

Remarks The values referenced by x and y will be summed and the result will be stored in the location pointed to by z. If an overflow occurs, the result will be ±Inf. If an underflow occurs, the result will be zero (0).

Example This example adds the data in res1 to the data in res2. The result is stored in the resia variable.

```
double res1, res2, resia;
.
.
measv(SMU1, &res1);/* Measure SMU1 voltage; store */
/* in res1. */
measi(SMU2, &res2);/* Measure SMU2 current; store */
/* in res2. */
kfpadd(&res1, &res2, &resia);/* Adds res1 and res2; return */
/* result to resia. */
```

.
.

kfpdiv Keithley floating point divide

Purpose Divides two real numbers and stores the result in a specified variable.

Format

```
int kfpdiv(double *x, double *y, double *z);
```

x	The dividend.
y	The divisor.
z	A variable where the result of x/y will be stored.

Remarks The value referenced by `x` will be divided by the value referenced by `y` , and the result will be stored in the location pointed to by `z`. If an overflow occurs, the result will be `±Inf`. If an underflow occurs, the result will be zero (0).

Example This example divides the data in `res1` by the data in `res2`. The result is stored in the `resia` variable.

```
double res1, res2, resia;
.
.
measv(SMU1, &res1);/* Measure SMU1 voltage; store */
/* in res1. */
measi(SMU2, &res2);/* Measure SMU2 current; store */
/* in res2. */
kfpdiv(&res1, &res2, &resia);/* Divide res1 by res2; return */
/* result to resia. */
.
.
```

kfpexp Keithley floating point exponential

Purpose Supplies the base of natural logarithms (e) raised to a specified power and stores the result as a variable.

Format

```
int kfpexp(double *x, double *z);
```

`x`	The exponent.
`z`	The variable where the result of e^x will be stored.

Remarks e raised to the power of the value referenced by `x` will be stored in the location pointed to by `z`. If an overflow occurs, the result will be `±Inf`. If an underflow occurs, the result will be zero (0).

Example In this example, `kfpexp` raises the base of natural logarithms to the power specified by the
exponent `res4`. The result is stored in `res4e`.

```
double res4, res4e;
.
.
measv(SMU1, &res4);/* Raise the base of natural */
/* logarithms e to the power */
/* res4; */
kfpexp(&res4, &res4e;/* return the result to res4e. */
.
.
```

kfplog Keithley floating point logarithm

Purpose Returns the natural logarithm of a real number to the specified variable.

Format

```
int kfplog(double *x, double *z);
```

`x`	A variable containing a floating point number.
`z`	A variable where the result of ln (`x`) will be stored.

Remarks This function returns a natural logarithm, not a common logarithm. The natural logarithm of the value referenced by `x` will be stored in the location pointed to by `z`.

If a negative value or zero (0) is supplied for `x`, a log of negative value or zero (0) error will be generated and the result will be `NaN` (not a number).

Example This example calculates the natural logarithm of a real number (`res1`). The result is stored in `logres`.

```
double res1, logres;
.
.
measv(SMU1, &res1);/* Measure SMU1; store in res1. */
kfplog(&res1, &logres);/* Convert res1 to a natural */
/* LOG and store in logres. */
.
```

kfpmul Keithley floating point multiply

Purpose Multiplies two real numbers and stores the result as a specified variable.

Format

```
int kfpmul(double *x, double *y, double *z);
```

x A variable containing the multiplicand.

y A variable containing the multiplier.

z The variable where the result of `x` *`y` will be stored.

Remarks The value referenced by `x` will be multiplied by the value referenced by `y` , and the result will be stored in the location pointed to by `z`. If an overflow occurs, the result will be `±Inf`. If an underflow occurs, the result will be zero (0).

Example This example multiplies variables `res1` and `res2`. The result is stored in the variable `pwr2`.

```
double res1, res2, pwr2;
.
.
measi(SMU1, &res1);/* Measure SMU1 current; */
/* store in res1. */
measv(SMU1, &res2);/* Measure SMU1 voltage; */
/* store in res2. */
kfpmul(&res1, &res2, &pwr2);/* Multiply res1 by res2; */
/* return result to pwr2. */
.
.
```

kfpneg Keithley floating point negative value

Purpose Changes the sign of a value and stores the result as a specified variable.

Format

```
int kfpneg(double *x, double *z);
```

x A variable containing the number to be converted.

z A variable where the result of -`x` will be stored.

Remarks If the value is positive, it is converted to a negative; if the value is negative, it is converted to a positive.

Example This example changes the sign of a positive voltage reading. `forcev` outputs a positive 10 V from SMU1. The current is measured with `measi`, and the result is stored as `res4`. `kfpneg` reads `res4` and converts the data to a negative value. `res4` is then overwritten with the converted value.

```
double res4;
.
.
forcev(SMU1, 10.0);/* Output 10 V from SMU1. */
measi(SMU1, &res4);/* Measure SMU1 current; store */
/* in res4. */
kfpneg(&res4, &res4);/* Convert sign of res4; */
./* return results to res4. */
.
```

kfppwr Keithley floating point power

Purpose Raises a real number to a specified power and assigns the result to a specified variable.

Format

```
int kfppwr(double *x, double *y, double *z);
```

x	A variable containing a floating point number.
y	A variable containing the exponent.
z	A variable where the result of x^y will be stored.

Remarks The value referenced by `x` will be raised to the power of the value referenced by `y`, and the result will be stored in the location pointed to by *z*. If an overflow occurs, the result will be `±Inf`. If an underflow occurs, the result will be zero (0).

If `x` points to a negative number, a power of a negative number error will be generated, and the result returned will be `-Inf`.

If `x` points to a value of zero (0) and `y` points to a negative number, a divide by zero (0) error will be generated, and the result returned will be `+Inf`.

If `x` points to a value of 1.0, the result will be 1.0, regardless of the exponent.

Example The following example raises the variable `res2` by the power of three. The result is stored in `pwres2`.

```
double res2, pwres2, power=3.0;
.
.
measv(SMU1, &res2);/* Measure SMU1; store */
/* result in res2. */
kfppwr(&res2, &power,
 &pwres2);/* res2 to the third power; */
/* return result to pwres2. */
.
```

kfpsqrt Keithley floating point square root

Purpose Performs a square root operation on a real number and returns the result to the specified variable.

Format

```
int kfpsqrt(double *x, double *z);
```

`x` A variable containing a floating point number.

`z` A variable where the result, the square root of `x`, will be stored.

Remarks

The square root of the value referenced by `x` will be stored in the location pointed to by `z`.

If `x` points to a negative number, a square root of negative number error will be generated, and the result will be `NaN` (not a number).

Example

This example converts a real number (`res1`) into its square root. The result is stored in `sqres2`.

```
double res1, sqres2;
.
.
measv(SMU1, &res1);/* Measure SMU1; store result */
./* in res1. */
kfpsqrt(&res1, &sqres2);/* Find square root of res1; */
/* return result to sqres2. */
.
```

kfpsub Keithley floating point subtract

Purpose

Subtracts two real numbers and stores their difference in a specified variable.

Format

```
int kfpsub(double *x, double *y, double *z);
```

x A variable containing the minuend.

y A variable containing the subtrahend.

z The variable where the result of *x* – *y* will be stored.

Remarks

The value referenced by *y* will be subtracted from the value referenced by *x*, and the result will be stored in the location pointed to by *z*. If an overflow occurs, the result will be `±Inf`. If an underflow occurs, the result will be zero (0).

Example

This example subtracts `res2` from `res1`. The result is returned to `diff2`.

```
double res1, res2, diff2;
.
.
measv(SMU1, &res1);/* Measure SMU1; store result */
/* in res1. */
measv(SMU2, &res2);/* Measure SMU2; store result */
/* in res2. */
kfpsub(&res1, &res2, &diff2);/* Subtract res2 from res1; */
./* return the place with */
/* result to diff2. */
.
```

kibcmd Keithley GPIB command

Purpose

Enables universal, addressed, and unaddressed GPIB bus commands to be sent through the GPIB interface. These commands can consist of any command that is valid

with the ATN line asserted, such as `DCL`, `SDC`, `GET`, and so on. The following table lists these GPIB commands.

Format

```
int kibcmd(unsigned int timeout, unsigned int numbytes, char*
cmdbuffer);
```

timeout The timeout for transfer (in 100 ms ticks).

numbytes The number of BYTES in `cmdbuffer` to send with the ATN line asserted.

`cmdbuffer` The array containing the BYTES to transfer over the GPIB interface.

Table 8-15
GPIB command list

GPIB command	Data byte (Hex)	Comments
Universal		
(local lockout)	11	Locks-out front panel controls.
(device clear)	14	Returns instrument to default conditions.
(serial poll enable)	18	Enables serial polling.
(serial poll disable)	19	Disables serial polling.
Addressed		
(selective device clear)	04	Returns instrument to default conditions.
(go to local)	01	Sends go to local.
(group execute trigger)	08	Triggers instrument for reading.
Un-addressed		
(unlisten)	3F	Removes all listeners from GPIB bus.
(untalk)	5F	Removes any talkers from GPIB bus.
(listen address group)	20-3E	Place instrument at this primary address (0 through 30) in listen mode.
(talk address group)	40-5E	Place instrument at this primary address (0 through 30) in talk mode.
(secondary command group)	60-7E	Place instrument at this secondary address (0 through 30) in listen mode.

Remarks

`kibcmd` performs the following steps:

1. Asserts attention (ATN).
2. Sends byte string (command buffer).
3. De-asserts ATN.

Example

This example illustrates how the `kibcmd` command could be used to issue a GPIB bus trigger command to a GPIB instrument located at address 15:

```
int status;
char GPIBtrigger[5] = {0x3F, 0x2F, 0x08, 0x3F, 0x00};
/* Unlisten = 3F (UNL) */
/* Listen address = 32 + 15 = 2F */
/* Group Execute Trigger (GET) = 08 */
/* UNL */
/* Terminate string with NULL */
.
.
.
```

```
status = kibcmd(30, strlen(GPIBtrigger),GPIBtrigger);
/* Use 3s timeout */
```

kibdefclr Keithley GPIB define device clear

Purpose

Defines the device-dependent command sent to an instrument connected to the GPIB interface. This string is sent during any normal tester-based `devclr`. It ensures that if the tester is calling `devclr` internally, any external GPIB device will be cleared with the given string.

Format

```
int kibdefclr(int pri_addr, int sec_addr, unsigned int timeout,
double delay, unsigned int snd_size, char *sndbuffer);
```

pri_addr The primary address of the instrument; numbers 1 through 30 are valid (the controller uses address 31).

sec_addr The secondary address of the instrument; numbers 1 through 30 are valid. If the instrument device does not support secondary addressing, this parameter must be -1.

timeout The GPIB timeout for the transfer. This timeout is in 100 ms units (for example, timeout = 40 = 4.0 s).

delay The time to wait after the device-dependent string is sent to the device, in seconds.

snd_size The number of bytes to send over the GPIB interface.

sndbuffer The physical byte buffer containing the data to send over the bus. This is the physical CLEAR string. A maximum of 1024 bytes are allowed.

Remarks

Each call to `kibdefclr` copies parameters into a data structure within the tester memory. These data structures are allocated dynamically. After the execution of the command buffer using `execut`, these tables are cleared. Any strings previously defined MUST be redefined.

The tester system allows you to define a maximum of 20 clear and 20 initialization strings. Each string may contain up to a maximum of 1024 bytes. Once defined, these strings remain in effect until the `execut` statement is processed.

Strings are sent over the GPIB interface in a first-in, first-out queue. This means that the first call to `kibdefclr` or `kibdefint` will be the first string sent over the GPIB. `devclr` (`kibdefclr`) strings are ALWAYS sent prior to initialization.

The KIBLIB `devclr` strings are sent PRIOR to `devclr` and `devint` execution. This may be a problem when communicating with any Keithley supported GPIB instruments. This may also have an effect on `bsweep`, because `bsweep` issues a `devclr` to clear active sources. It is not recommended to use GPIB instruments when performing `bsweep` tests.

kibdefdelete Delete GPIB definition strings for devclr and devint

Deletes all command definitions previously made with the `kibdefclr` (Keithley GPIB define device clear) and `kibdefint` (Keithley GPIB define device initialize) commands. Once this command is issued, any previous definitions made using `kibdefclr` or `kibdefint` will no longer occur at `devint` or `devclr` time.

Format

```
int kibdefdelete( void );
```

Remarks This function can be overridden by re-issuing the `kibdefint` and `kibdefclr` commands.

kibdefint Keithley GPIB define device initialize

Purpose Defines a device-dependent command sent to an instrument connected to the GPIB interface. This string is sent during any normal tester-based `devint`. It ensures that if the tester is calling `devint` internally, any external GPIB device will now be initialized along with the rest of the known instruments.

Format

```
int kibdefint(int pri_addr, int sec_addr, unsigned int timeout,
double delay, unsigned int snd_size, char *snd_buff);
```

`pri_addr`	The primary address of the instrument; 1 through 30 is valid (the GPIB uses address 31).
`sec_addr`	The secondary address of the instrument; 1 through 30 is valid. If the instrument device does not support secondary addressing, this parameter must be -1.
`timeout`	The GPIB transfer timeout. This timeout is in 100 ms units (for example timeout = 40 = 4.0 s).
`delay`	The time to wait after the device-dependent string is sent to the device, in seconds.
`snd_size`	The number of bytes to send over the GPIB interface.
`snd_buff`	The physical byte buffer containing the data to send over the bus. This is the INITIALIZE string. A maximum of 1024 bytes is allowed.

Remarks Each call to `kibdefclr` and `kibdefint` copies parameters to a data structure within tester memory. These data structures are allocated dynamically. After the execution of the command buffer using `execut`, these tables are cleared, and any strings previously defined MUST be redefined.

The tester system lets you define a maximum of 20 clear and 20 initialization strings. Each string may contain up to a maximum of 1024 bytes. Once defined, these strings remain in effect until the `execut` statement is executed.

Strings are sent over the GPIB in a first-in, first-out queue. This means that the first call to `kibdefclr` or `kibdefint` will be the first string sent over the GPIB interface. `devclr` (`kibdefclr`) strings are ALWAYS sent prior to initialization.

All `kiblib devclr` and `devint` strings are sent PRIOR to `devclr` and `devint` execution. This may be a problem when communicating with any Keithley-supported GPIB instruments. This may also have an effect on `bsweep`, because `bsweep` issues a `devclr` to clear ALL active sources. It is not recommended to use GPIB instruments when performing `bsweep` tests.

kibrcv Keithley GPIB receive

Purpose Used to read a device-dependent string from an instrument connected to the GPIB interface.

Format

```
int kibrcv(int pri_addr, int sec_addr, char term, unsigned int
timeout, unsigned int rcv_size, unsigned int *rcv_len, char
*rcv_buff);
```

`pri_addr`	The primary address of the instrument; 1 through 30 is valid (the GPIB controller uses address 31).
`sec_addr`	The secondary address of the instrument; 1 through 30 is valid. If the instrument device does not support secondary addressing, this parameter must be -1.
`term`	The ASCII delimiter character of the returned string. This is the byte used for terminating data buffer read.
`timeout`	The GPIB transfer timeout. This timeout is in 100 ms units (for example, timeout = 40 = 4.0 s).
`rcv_size`	The physical receive buffer size. This is the maximum number of bytes that can be read from the device.
`rcv_len`	The number of bytes that are read from the device on the GPIB interface. This variable is returned by the tester after all bytes are read from the device.
`rcv_buff`	The physical BYTE buffer destined to receive the data from the device connected to the GPIB interface.

Remarks

`kibrcv` receives a buffer from the GPIB interface by performing the following steps:

1) Assert attention (ATN).
2) Send device LISTEN address.
3) Send device TALK address.
4) Send secondary address (if not -1).
5) De-assert ATN.
6) Read byte array from device `Rcv_Buff` until end-or-identify (EOI) or the delimiter is received.
7) Assert ATN.
8) Send UNTalk (UNT).
9) Send UNListen (UNL).
10) De-assert ATN.

`kibrcv`: `rcv_size` defines the maximum number of bytes physically allowed in the buffer. If `rcv_size` is greater than the byte string returned by the instrument, then the device is short-cycled and only the maximum number of bytes is returned.

kibsnd Keithley GPIB send

Purpose

Sends a device-dependent command to an instrument connected to the GPIB interface.

Format

```
int kibsnd(int pri_addr, int sec_addr, unsigned int timeout,
unsigned int send_len, char *send_buff);
```

`pri_addr`	The primary address of the instrument; 1 through 30 is valid. (The controller uses GPIB address 31).
`sec_addr`	The secondary address of the instrument; 1 through 30 is valid. If the instrument device does not support secondary addressing, this parameter must be -1.
`timeout`	The GPIB transfer timeout. This timeout is in 100 ms units (for example, timeout = 40 = 4.0 s).
`send_len`	The number of bytes to send over the GPIB interface.

`send_buff`	The physical byte buffer containing the data to send over the bus.

Remarks `kibsnd` sends a buffer out the GPIB interface by performing the following steps:

1) Assert attention (ATN).
2) Send device LISTEN address.
3) Send secondary address (if not -1).
4) Send my TALK address.
5) De-assert ATN.
6) Send `Send_Buff` with end-or-identify (EOI) asserted with the LAST BYTE.
7) Assert ATN.
8) Send UNTalk (UNT).
9) Send UNListen (UNL).
10) De-assert ATN.

kibspl Keithley GPIB serial poll

Purpose Serial polls an instrument connected to the GPIB interface.

Format

```
int kibspl(int pri_addr, int sec_addr, unsigned int timeout,
int *statusbyte);
```

`pri_addr`	The primary address of the instrument; 1 through 30 is valid (the controller uses GPIB address 31).
`sec_addr`	The secondary address of the instrument; 1 through 30 is valid. If the instrument device does not support secondary addressing, this parameter must be -1.
`timeout`	The GPIB polling timeout. This timeout is in l00 ms units (for example, timeout = 40 = 4.0 s).
`statusbyte`	The serial poll status byte returned by the device presently being polled. The `statusbyte` variable must be an integer.

Remarks `kibspl` performs the following steps:

1) Assert attention (ATN).
2) Send serial poll enable (SPE).
3) Send LISTEN address.
4) Send device TALK address.
5) Send secondary address (if not -1).
6) De-assert ATN.
7) Poll GPIB interface until data is available.
8) Read Serial_Poll_BYTE from device (if data is available), else.
9) Serial_Poll_BYTE = 0 (indicating error; device not SRQing).
10) Assert ATN.
11) Send serial poll disable (SPD).
12) Send UNTalk (UNT).
13) Send UNListen (UNL).
14) De-assert ATN.

kibsplw Keithley GPIB

Purpose Used to synchronously serial poll an instrument connected to the GPIB interface. This command waits for SRQ to be asserted on the GPIB by any device. After SRQ is asserted, a serial poll sequence is initiated for the device and the serial poll status byte is returned.

Format

```
int kibsplw(int pri_addr, int sec_addr, unsigned int timeout,
int *statusbyte);
```

`pri_addr`	The primary address of the instrument; 2 through 31 is valid.
`sec_addr`	The secondary address of the instrument; 1 through 31 is valid. If the instrument device does not support secondary addressing, this parameter must be -1.
`timeout`	The GPIB polling timeout. The timeout is in l00 ms units (for example, timeout = 40 = 4.0 s).
`stausbyte`	The serial poll status byte variable name returned by the device presently being polled.

Remarks `kibsplw` performs the following steps:

1) Wait with timeout for general SRQ assertion on the GPIB.
2) Call `kibspl`.

kspcfg Configure the port

Purpose Configures and allocates a serial port for RS-232 communications.

Format

```
int kspcfg(int port, int baud, int databits, int parity, int
stopbits, int flowctl);
```

`port`	The RS-232 port to be used. Currently only port 1 is supported.
`baud`	The transmission rate that will be used. Valid rates are: 2400, 4800, 9600, 14400, and 19200 baud.
`databits`	The number of data bits that will be used. Valid inputs are 7 or 8 bits.
`parity`	Determines whether or not parity bits will be transmitted. Valid inputs are: 0 (no parity), 1 (odd parity), or 2 (even parity).
`stopbits`	Sets the number of stop bits to be transmitted. Valid inputs are: 1 or 2.
`flowctl`	Determines the type of flow control that will be used. Valid inputs are: 0 (no flow control), 1 (XON/XOFF flow control), or 2 (hardware).

Remarks Port 1 must not be allocated to another program or utility when using the `ksp` (Keithley Serial Port) commands.

- The `databits`, `parity`, `stopbits`, and `flowctl` settings must match those on the instrument or device that you wish to control.
- Using a flow control setting of 0 may result in buffer overruns if the device or instrument that you are controlling has a high data rate.
- If you use a flow-control setting of 2 (hardware), you must make sure that the RS-232 cable has enough wires to handle the RTS/CTS signals.

Example Here is an example of how you would use `kspcfg` to set port 1 to 19200 baud, 8 data bits, odd parity, 1 stop bit, and xon/xoff flow control:

```
int status;
.
.
.
status = kspcfg(1, 19200, 8, 1, 1, 1);/* port 1, 19200 baud,
   8 bits, odd parity,
   1 stop bit, and
   xon-xoff flow ctl */
```

kspdefclr Define string to clear RS-232 instrument on devclr

Purpose Defines a device-dependent character string sent to an instrument connected to a serial port. This string is sent during the normal tester `devlcr` process. It ensures that if the tester is calling `devclr` internally, any device connected to the configured serial port will be cleared with the given string.

Format

```
int kspdefclr(int port, double timeout, double delay, int
buffsize, char *buffer);
```

port	The RS-232 port to be used. Currently only port 1 is supported. This port must have been previously configured for communications with the `kspcfg` command.
timeout	The serial communications timeout. The valid input range is 0 to 600 seconds.
delay	The amount of time to delay after sending the string to the serial device. The valid input range is 0 to 600 seconds.
buffsize	The length of the string to send to the serial device.
buffer	A character string containing the data to send to the serial device.

Remarks Before issuing this command, you must configure the serial port using the `kspcfg` command.

- The commands sent to the serial device are issued in the order in which they were defined using the `kspdefclr` command.
- The `kspdefdelete` command can be used to delete any previous definitions.
- The `kspdefclr` and `kspdefint` command strings are sent prior to normal (for example, an SMU) instrument `devclr` and `devint` execution.

kspdefdelete Delete RS-232 definition strings for devclr and devint

Purpose Deletes all command definitions previously made with the `kspdefclr` (Keithley Serial Define Device Clear) and `kspdefint` (Keithley Serial Define Device Initialize) commands. Once this command is issued, any previous definitions made using `kspdefclr` or `kspdefint` will no longer occur at `devint` or `devclr` time.

Format

```
int kspdefdelete( void );
```

Remarks This function can be overridden by re-issuing the original `kspdefint` and `kspdefclr` commands.

kspdefint Define string to clear RS-232 instrument on devint

Purpose :Defines a device-dependent character string sent to an instrument connected to a serial port. This string is sent during the normal tester `devint` process. It ensures that if the tester is calling `devint` internally, any device connected to the configured serial port will be cleared with the given string.

Format

```
int kspdefint(int port, double timeout, double delay, int
buffsize, char *buffer);
```

port The RS-232 port to be used. Currently only port 1 is supported. This port must have been previously configured for communications with the `kspcfg` command.

timeout The serial communications timeout. The valid input range is 0 to 600 seconds.

delay The amount of time to delay after sending the string to the serial device. The valid input range is 0 to 600 seconds.

buffsize The length of the string to send to the serial device.

buffer A character string containing the data to send to the serial device.

Remarks Before issuing this command, you must configure the serial port using the `kspcfg` command.

- The commands sent to the serial device are issued in the order in which they were defined using the `kspdefclr` command.
- The `kspdefdelete` command can be used to delete any previous definitions.
- The `kspdefclr` and `kspdefint` command strings are sent prior to normal (for example, an SMU) instrument `devclr` and `devint` execution.

ksprcv Receive device-dependent command string

Purpose Used to read data from an instrument connected to a serial port.

Format

```
int ksprcv(int port, char terminator, double timeout, int
rcvsize, int *rcv_len, char *rcv_buffer);
```

port The RS-232 port to be used. Currently only port 1 is supported. This port must have been previously configured for communications with the `kspcfg` command.

terminator The ASCII terminator for the received data. This character is used to terminate the read.

timeout The serial communications timeout. The valid input range is 0 to 600 seconds.

rcvsize The physical buffer size. This is used to control the maximum number of characters that can be read from the device.

rcv_len The actual number of characters read from the device. This value is returned to the `ksprcv` command by the software.

rcv_buffer A character array in which to store the data returned from the serial device.

kspsnd Send device-dependent string

Purpose Sends a device-dependent command to an instrument attached to a RS-232 serial port.

Format

```
int kspsnd( int port, double timeout, int cmdlen, char *cmd);
```

port The RS-232 port to be used. Currently only port 1 is supported. This port must have been previously configured for communications with the `kspcfg` command.

timeout The serial communications timeout. The valid input range is 0 to 600 seconds.

cmdlen The number of characters that you are sending out the serial port.

cmd The character array containing the data that you want sent out of the serial port.

limitX Limit a voltage or current

Purpose Allows the programmer to specify a current or voltage limit other than the instrument's default limit.

Format

```
int limiti(int instr_id, double limit_val);
int limitv(int instr_id, double limit_val);
```

inst_id The instrument identification code of the instrument on which to impose a source value limit.

limit_val The maximum level of the current or voltage. The value is bidirectional. For example, a `limitv (SMU1, 10.0)` limits the voltage of the current source SMU1 to ±10.0 V. A `limiti (SMU1, 1.5E-3)` limits the current of the voltage source SMU1 to ±I.5 mA.

Remarks Use `limiti` to limit the current of a voltage source. Use `limitv` to limit the voltage of a current source.

NOTE *If the instrument is ranged below the programmed limit value, the instrument will temporarily limit to full scale of range.*

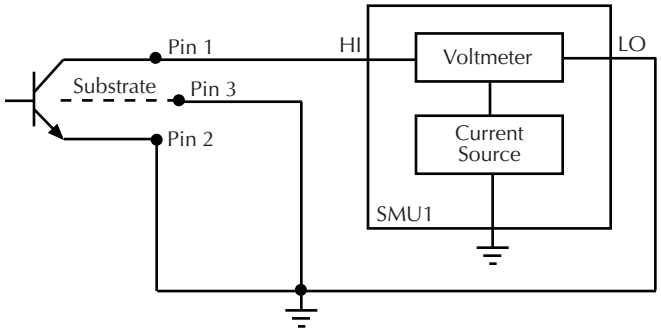
This function must be called in the test sequence before the associated `forceX`, `pulseX`, `bsweepx`, `sweepX`, or `searchX` function is used to generate the voltage or current. `limitX` also sets the top measurement range of an autoranged measurement.

The limits set within a particular test sequence are cleared when `devint` or `execut` are called.

If you need a voltage limit greater than 20 V at an SMU that has been set to force zero current, use a `measv` call to set the SMU to autorange to a higher range, OR use `rangev` to set a higher voltage range. Similarly, if you need a current limit of greater than 10 mA at an SMU that has been set to force zero volts, use a `measi` call to set the SMU to autorange to a higher range OR use **rangev** to set a higher current range.

Example This example measures the breakdown voltage of a device. The limit is set at 150.0 V. This limit is necessary to override the default limit of the SMU, which would otherwise be in effect.

Figure 8-97
Measuring device breakdown voltage




```
double ibceo, vbceo;
.
.
conpin(2, 3, GND, 0);
conpin(SMU1, 1, 0);
limitv(SMU1, 150.0);/* Limit voltage at 150 V. */
forcei(SMU1, ibceo);/* Force current through DUT. */
measv(SMU1, &vbceo);/* Measure breakdown voltage; */
.                        /* return results to vbceo. */
.
```

lorangeX Select bottom range

Purpose Defines the bottom autorange limit.

Format

```
int lorangei(int inst_id, double range);
int lorangev(int inst_id, double range);
```

inst_id The instrument identification code.

range The value of the desired instrument range, in volts or amperes.

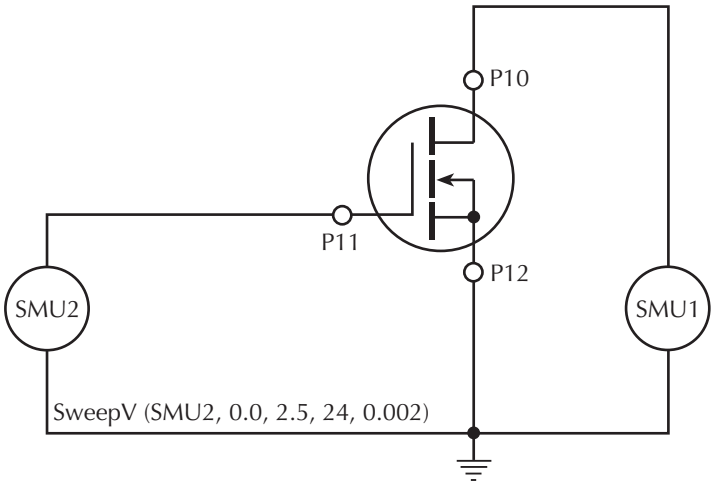
Remarks `lorange` is used with autoranging to limit the number of range changes, and thus saves test time.

If the instrument was on a range lower than the one specified by `lorange`, the range is changed. Model 4200-SCS automatically provides any range change settling delay that may be necessary due to this potential range change.

Once defined, `lorange` is in effect until a `devclr`, `devint`, `execut`, or another `lorangeX` executes.

Example This example illustrates how you would select the bottom autorange limit.

Figure 8-98
Defining bottom autorange limit




```
double idatrg [25];
.
.
conpin(SMU1, 10, 0);
conpin(SMU2, 11, 0):
conpin(12, GND, 0);
lorangei(SMU1, 2.0E-6);/* Select 2 mA as minimum */
                        /* range during autoranging. */
smeasi(SMU1, idatvg);/* Setup sweep measurement */
                        /* of IDS. */
sweepv(SMU2, 0.0, 2.5, 24,
 0.002);/* Sweep gate from 0 to */
/* 2.5 V. */
```

measX Measure

Purpose :Allows the measurement of voltage, current, or time.

Format

```
int measi(int inst_id, double *result);
int meast(int inst_id, double *result);
int measv(int inst_id, double *result);
```

inst_id The instrument identification code.

result The variable assigned to the result of the measurement.

Remarks For a measurement conversion, the signal is sampled for a specific period of time. This sampling time for measurement is called the integration time. For the `measX` function, the integration time is fixed at 0.01 PLC. For 60 Hz line power, 0.01 PLC = 166.67 µs (0.01 PLC/60 Hz). For 50 Hz line power, 0.01 PLC = 200 µs (0.01 PLC/50 Hz).

NOTE *The only difference between* `measX` *and* `intgX` *is the integration time. For* `measX`*, the integration time is fixed at 0.01 PLC. For* `intgX`*, the default integration time is 1 PLC, but can set to any PLC value between 0.01 and 10.0.*

After the function is called, all relay matrix connections remain closed, and the sources continue to generate voltage or current. For this reason, two or more measurements can be made in sequence.

`rangeX` directly affects the operation of the `measX` function. The use of `rangeX` prevents the instrument addressed from automatically changing ranges when `measX` is called. This can result in an overrange condition such that would occur when measuring 10.0 V on a 4.0 V range. An overrange condition returns the value `1.0E+22` as the result of the measurement.

If used, `rangeX` must be located in the test sequence before the associated `measX` function.

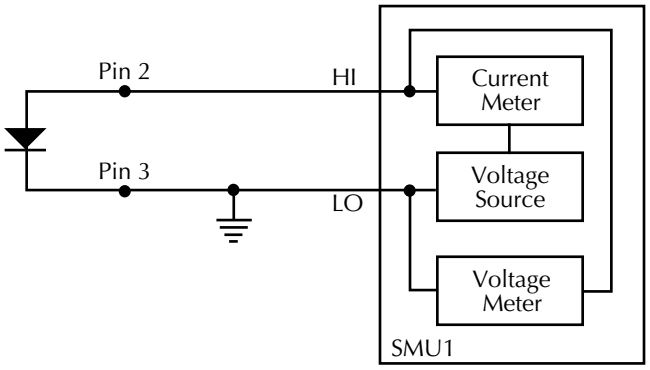
All measurements except `meast` invoke a timer snapshot measurement to be taken by all enabled timers. This timer snapshot can then be read with the `meast` command.

In general, measurement functions which return multiple results are more efficient than performing multiple measurement functions.

Compliance limits: A compliance limit setting goes into effect when the SMU is on a measure range that can accommodate the limit value. For manual ranging, the `rangeX` function is used to select the range. For autoranging, the `measi` or `measv` function will trigger a needed range change before the measurement is performed. See Using source compliance limits for details.

Example In this example, the diode's forward bias voltage is obtained from a single SMU.

Figure 8-99
measX




```
double if46, vf47;
.
.
if46 = 50e-3;
.
.
conpin(3, GND, 0);
conpin(SMU1, 2, 0);
forcei(SMU1, if46);/* Forward bias the diode; */
/* set SMU current */
/* limit to 50 mA. */
measv(SMU1, &vf47);/* Measure forward bias; */
/* return result to vf47. */
```

mpulse Measure pulse

Purpose This function uses an SMU to force a voltage pulse and measures both the voltage and current for exact device loading.

Format

```
int mpulse(int inst_id, double pulse_amplitude,double
pulse_duration, double *v_meas, double *i_meas);
```

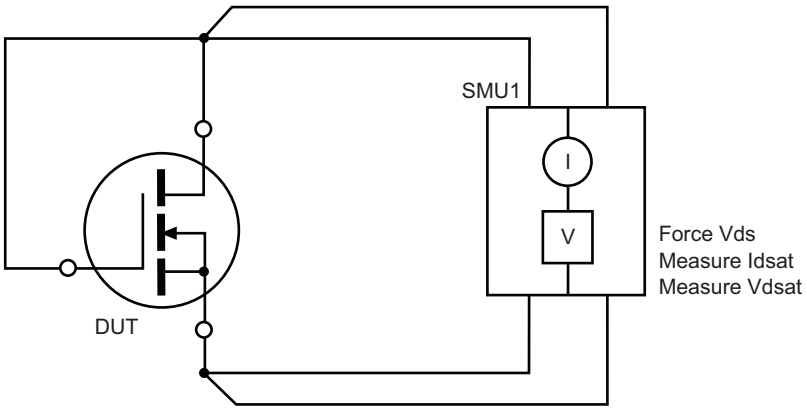
inst_id	The name of the instrument under control.
pulse_amplitude	The pulse height in volts.
pulse_duration	The pulse width in seconds. The measurements will be taken at the end of the pulse before the `mpulse` is shutdown.
v_meas	The variable used to receive the voltage on the output of the instrument at the time the pulse terminates.
i_meas	The variable used to receive the current drawn from the instrument. This measurement is taken simultaneously with the voltage, so the combined values are an exact representation of the device load at pulse termination.

Remarks Voltage and current are measured just before the pulse terminates.

Pulsing is useful for devices that exhibit self-heating, which could damage the device or shift operating characteristics. Examples are high-power GaAs transistors or BJTs, but even some silicon devices exhibit self-heating.

Example The following example measures the drain current of a MOSFET when Vds equals Vgs. A voltage pulse, Vds, is applied to the drain. The pulse duration is 1 ms. Voltage across the MOS transistor, Vdsat, and drain current, Idsat, are measured.

Figure 8-100
mpulse




```
double vdsat, idsat, vds;
.
.
mpulse(SMU1, vds, 1.0E-3,/* Pulse output of SMU1. */
 &vdsat, &idsat);
```

pulseX Pulse of a voltage or current

Purpose This function directs a SMU to force a voltage or current at a specific level for a predetermined length of time.

Format

```
int pulsei(int inst_id, double forceval, double time);
int pulsev(int inst_id, double forceval, double time);
```

inst_id The instrument identification code.

forceval — The level of voltage or current in volts or amperes, respectively, to be forced. The value can be positive or negative. For example, a `pulsev (SMU1, 10.0, 10 e-3)` generates +10.0 V for 10 ms, and a `pulsei (SMU1, -1.5E-3, 10 e-3)` generates -1.5 mA for 10 ms.

time — The pulse duration in seconds. For example, a time of 0.5 initiates a time of 0.5 s, and a time of 2.0E-2 initiates a time of 20 ms. The minimum practical time for an SMU source is dependent on the voltage or current level being sourced and the impedance of the DUT.

The ranges of current and voltage available vary with the specific instrument type. For more detailed information, refer to the specific hardware manuals.

Remarks

After `pulseX` is executed, the output is turned off. In order to perform measurements, the output must be turned back on. `measX` can measure:

- Residual voltage or current as it decays after removal of the initial application.
- Capacitance between DUT pins as the residual voltage or current decays.

All measurements performed using the `pulseX` and `measX` functions are performed AFTER the pulse has completed.

NOTE *When the source is not operating, measurements are not allowed.*

Whenever `pulseX` is executed, either a default or a programmed current or voltage limit is in effect. Refer to the limit command for additional information.

When using `limitX`, `rangeX`, and `pulseX` on the same source at the same time in a test sequence, call `limitX`, then `rangeX`, then `pulseX`.

Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

Example

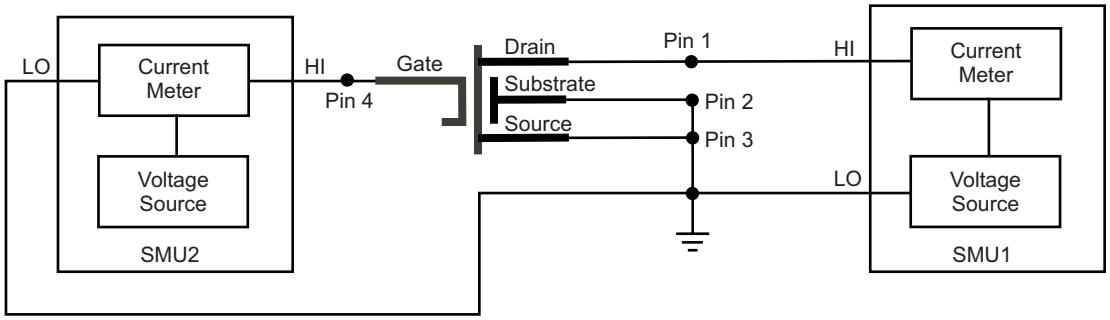
Here the threshold voltage shift of an FET is measured by performing two `searchv` functions, as follows:

1) `searchv` measures the gate voltage required to initiate a drain current of 10 μA.

2) `searchv` measures the gate voltage required to initiate a drain current of 10 μA immediately after a 20 V pulse is applied to the gate.

Note that the second `searchv` was called without reprogramming `trigil`. This is possible because `clrtrg`, clear trigger, was not used.

Figure 8-101
pulseX




```
        double res1, res2;
        .
        .
        conpin(GND, 2, 3, 0);
        conpin(SMU1, 1, 0);
        conpin(SMU2, 4, 0);
        forcev(SMU1, .5);
        trigig(SMU1, +1.E-5);/* Set the trigger point for */
        /* 10mA. */
        searchv(SMU2, 0.0, 3.0, 7,/* Increase voltage until */
         2.0E-5, &res1);/* trigger point occurs. */
        /* Return results to res1. */
        pulsev(SMU2, 20.0, 5.E-4);/* Apply a 20 V pulse to the */
/* gate for 500ms. */
        searchv(SMU2, 0.0, 3.0, 7,/* Increase voltage until */
         2.0E-5, &res2);/* trigger point occurs. */
        /* Return results to res2. */
```

rangeX Select range

Purpose

Selects a range and prevents the selected instrument from autoranging. By selecting a range, the time required for autoranging is eliminated.

Format

```
int rangei(int inst_id, double range);
int rangev(int inst_id, double range);
```

inst_id The instrument identification code.

range The value of the highest measurement to be taken. The most appropriate range for this measurement will be selected. If `range` is set to 0, the instrument will autorange.

Remarks

`rangeX` is primarily intended to eliminate the time required by the automatic range selection performed by a measuring instrument. Because `rangeX` will prevent autoranging, an overrange condition can occur, for example, measuring 10.0 V on a 2.0 V range. The value `1.0E+22` is returned when this occurs.

`rangeX` can also reference a source, because an SMU can be either of the following:

- Simultaneously a voltage source, voltmeter, and current meter.
- Simultaneously a current source, current meter, and voltmeter.

The range of an SMU is the same for the sourcing function and the measuring function.

Compliance limits – When selecting a range below the limit value, whether it is explicitly programmed or the default value, an instrument will temporarily use the full scale value of the range as the limit. This will not change the programmed limit value and, if the

instrument range is restored to a value higher than the programmed limit value, the instrument will again use the programmed limit value. See Using source compliance limits earlier in this section for more information.

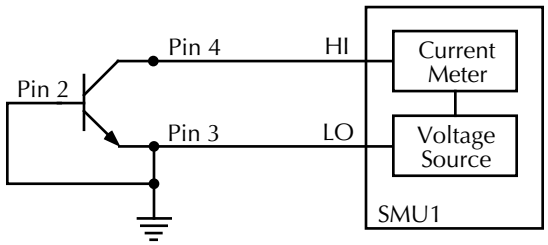
When changing the instrument range, be careful not to overrange the instrument. For example, a test initially performed in the 10 mA range with a 5 mA limit is changed to test in the 1 mA range with a 1 mA limit. Notice that the limit is lowered from 5 mA to 1 mA to avoid overranging the 1 mA setting.

Changing source mode may change measure range

When changing the source mode of the SMU, the measure range may change. This change minimizes variations in the SMU output level. The source mode of the SMU refers to its voltage sourcing or current sourcing capability. Changing the source mode means using a command (such as `forceX`) to change the SMU mode from forcing voltage to forcing current (or from forcing current to forcing voltage). For example, if the SMU is programmed to force voltage (`forcev`), and then is programmed with to force current (`forcei`), to ensure a consistent output signal the previously programmed current measure range may change. Make sure the desired measure range is set by sending the `rangeX` command after switching the source mode. The commands that can change the source mode are `asweepX`, `bsweepX`, `forceX`, `pulseX`, `searchX`, and `sweepX`.

Example

Figure 8-102
rangeX




```
double icer2;
.
.
conpin(3, 2, GND, 0);
conpin(SMU1, 4, 0);
limiti(SMU1, 1.0E-3);/* Limit current to 1.0 mA. */
rangei(SMU1, 2.0E-3);/* Select range for 2mA. */
forcev(SMU1, 35.0);/* Force 35 V. */
measi(SMU1, &icer2);/* Measure leakage; return */
/* results to icer2. */
```

rdelay Realtime delay

Purpose A user-programmable delay in seconds.

Format

```
int rdelay(double n);
```

n The desired delay duration in seconds.

Example The following example measures a variable capacitance diode's leakage current. SMU1 presets 60 V across the diode. The device is configured in reverse bias mode with the high side of SMU1 connected to the cathode. This type of diode has high-capacitance and

low-leakage current. Therefore, a 20 ms delay is added. After the delay, current through SMU1 is measured and stored in the variable `ir4`.

```
double ir4;
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 0);
forcev(SMU1, 60.0);/* Generate 60 V from SMU1. */
rdelay(0.02);/* Pause for 20 ms. */
measi(SMU1, &ir4);/* Measure current; return */
/* result to ir4. */
```

rtfary Return force array

Purpose

Returns the force array determined by the instrument action. This eliminates the need to calculate the forced array in the application.

Format

```
int rtfary(double *result);
```

result The floating point array where the force values will be stored.

Remarks

This function, when used in conjunction with one of the sweep routines, lets you determine the exact forced value for each point in the sweep.

When the test sequence is executed, the sweep function initiates the first step of the voltage or current sweep. The sweep then logs the force point that the buffer specified by `rtfary`.

Locate `rtfary` before the sweep. The number of data points returned by `rtfary` is determined by the number of force points generated by the sweep.

Example

Refer to the examples for the `smeasx` and `sweepx` functions.

savgX Sweep average

Purpose

Performs an averaging measurement for every point in a sweep.

Format

```
int savgi(int instr_id, double *results, long count, double
delay);
```

```
int savgv(int instr_id, double *results, long count, double
delay);
```

instr_id The measuring instrument's identification code.

results The floating point array where the results are stored.

count The number of measurements made at each point before the average is computed.

delay The time delay in seconds between each measurement within a given ramp step.

Remarks

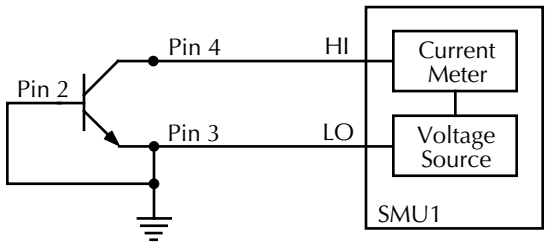
This function is used to create an entry in the measurement scan table. During any of the sweeping functions, a measurement scan is performed for every force point in the sweep. During each scan, a measurement will be made for every entry in the scan table. The measurements are made in the same order in which the entries were made in the scan table.

`savgX` sets up the new scan table entry to perform an averaging measurement. The measurement results will be stored in the array specified by results. Each time a measurement scan is made, a new measurement result will be stored at the next location in the results array. If the scan table is not cleared, performing multiple sweeps will simply keep adding new measurement results to the end of the array. Care must be taken to be sure the results array is large enough to hold all measurements which will be taken before the scan table is cleared. The scan table is cleared by an explicit call to `clrscn`, or implicitly when `devint` or `execut` is called.

When making each averaged measurement, *`count`* actual measurements will be made on the instrument *`delay`* seconds apart and the average calculated. This average is the value that will be stored in the results array.

Example This example obtains the measurement data needed to construct a graph showing the capacitance-versus-voltage characteristics of a variable capacitance diode. This diode is operated in reverse biased mode. SMU1 outputs a voltage that sweeps from 0 through -50 V. Capacitance is measured 26 times during the sweep. The results are stored in an array called `res1`.

Figure 8-103
savgX




```
double res1 [26];
.
.
conpin(3, 2, GND, 0);
conpin(SMU1, 4, 0);
savgi(SMU1, res1, 8, 1.0E-3);/* Measure average */
/* current 8 times per */
/* sample; return results to */
/* res1 array. */
sweepv(SMU1, 0.0, -50.0, 25,
 2.0E-2);/* Generate a voltage from 0 */
/* to -50 V over 25 steps.*/
```

scnmeas Scan measure

Purpose To perform a single measurement on multiple instruments at the same time.

Format

```
int scnmeas(void);
```

Remarks This function behaves like a single point sweep. It performs a single measurement on multiple instruments at the same time. Any forcing or delaying must be done prior to calling `scnmeas`.

`smeasX`, `sintgX`, or `savgX` must be used to set up result arrays just as is done for a sweep call. Each call to `scanmeas` will add one element to the end of each array.

Calls to `scnmeas` may be mixed with calls to `sweepX`, and all results will be appended to the result arrays in the same way multiple `sweepX` calls behave.

searchX Binary search measurement

Purpose

Used to determine the voltage or current required to obtain a desired current or voltage. It is useful in finding initial threshold points such as junction breakdown or transistor turn on.

Format

```
int searchi(int inst_id, double min_val, double max_val, long
iterate_no, double iterate_time, double *result);
```

```
int searchv(int inst_id, double min_val, double max_val, long
iterate_no, double iterate_time, double *result);
```

inst_id	The sourcing instrument's identification code.
min_val	The lower limit of the source range.
max_val	The upper limit of the source range.
iterat_no	The number of separate current or voltage levels to generate. The range of iterations is from 1 through 16.
iterate_time	The duration, in seconds, of each iteration.
result	The floating point variable assigned to the search operation result. It represents the voltage, with `searchv`, or current, with `searchi`, applied during the last search operation.

Remarks

`trigXg` or `trigXl` must be used with `search`. Triggers and `search` together initiate a search operation consisting of a series of steps referred to as iterations. During each iteration, the following events occur:

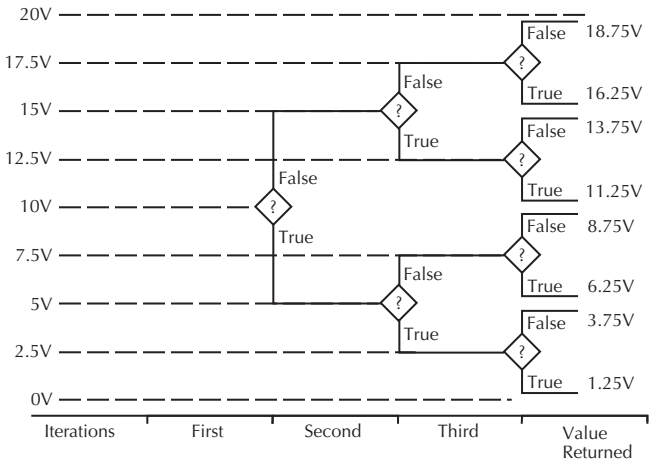
- A voltage or current is applied to a circuit node of the DUT.
- All triggers are evaluated.
- If the triggers evaluate true, the source value is moved toward the `min_val`. Otherwise, the source value is moved toward `max_val`. The source range is then divided in half for the next iteration.

Up to 16 iterations can be programmed. When all iterations are completed, a value of voltage or current is returned as the result of the search operation. This value is the voltage or current level required to match the trigger point.

The following example shows all binary search possibilities where the minimum and maximum source values are 0 and 20 V, respectively. Study the example and note the following:

- Three iterations, numbered one through three, are shown. Within a given iteration, the values of possible sourcing voltages are indicated.
- During the first iteration of the binary search process, 10 V is applied. This represents the midpoint of the minimum and maximum values.
- At the end of each iteration, the program determines whether to increase or decrease the source voltage. The determination is dependent on the evaluation of the trigger point.

Figure 8-104
searchX



The question mark (?) is the true or false determination.

As shown in the above figure, the true or false decision determines the voltage generated in the next step of the binary progression.

Because the function initiates a current or voltage from a source, its placement in a test sequence is critical. Therefore:

- Call `limitX` and `rangeX` before `searchX` when all three refer to the same instrument.
- Call `trigXg` or `trigXl` before `searchX`.

The search operation determines the source voltage or current required at one circuit node to generate a desired trigger point value at a second node. The resolution of the result depends on the number of iterations or steps and the actual current or voltage range being used by the instrument.

$$\frac{\text{voltage or current range}}{2^{(\text{iteration}+1)}}$$

For example, assume a source's minimum value and maximum value range is from 0 to 20 V, and the number of iterations is 16. The 20 V level automatically initiates an SMU 20 V sourcing range. Therefore, the resolution of the final source voltage returned is:

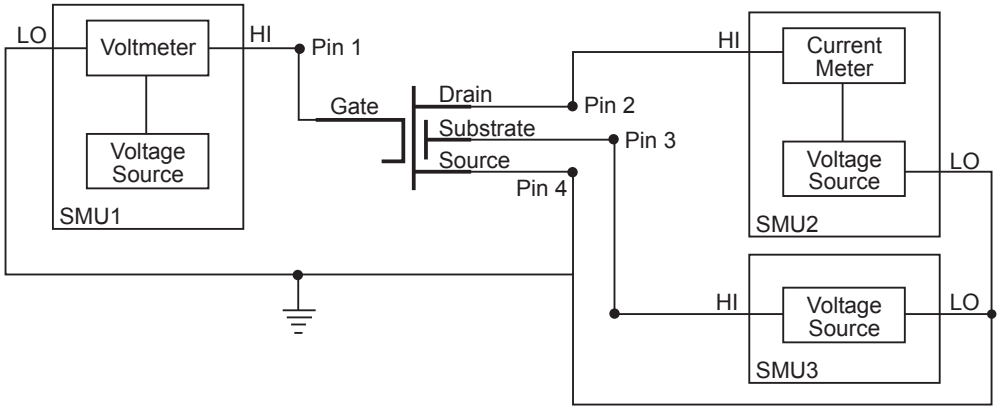
$$\frac{20}{2^{(16+1)}} = 1.2\text{mV}$$

Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

Example

The following example searches for the gate voltage required to generate a drain current of 1 μA. Eight separate gate voltages within the range of 0.6 V through 1.7 V are specified by `searchv`. After the eight iterations complete, the drain current is close to 1 μA, and the `searchv` operation is terminated. The gate voltage generated at this time by SMU1 is returned as the result `vgs1`.

Figure 8-105
searchv




```
double ssbiasv, vgs1, vds1;
.
conpin(SMU1, 1, 0);
conpin(SMU2, 2, 0);
conpin(SMU3, 3, 0);
conpin(GND, 4, 0);
trigig(SMU2, +l.0E-6);/* Set trigger point for 1µA. */
forcev(SMU3, ssbiasv);/* Apply a substrate bias */
                      /* voltage ssbiasv. */
forcev(SMU2, vds1);/* Apply a drain voltage of */
                      /* vds1. */
searchv(SMU1, 0.6, 1.7, 8, /* Set for 8 steps from 0.6 to */
 1.0E-3, &vgs1);/* 1.7V at 1ms.*/
                      /* per iteration; return the */
                      /* result to vgs1. */
```

setauto Re-enable Autoranging

Purpose

Re-enables autoranging and cancels any previous `rangeX` command for the specified instrument.

Format

```
int setauto(int inst_id);
```

`inst_id` The instrument identification code.

Remarks

When an instrument is returned to the autorange mode, it will remain in its present range for measurement purposes. The source range will change immediately.

Due to the dual mode operation of the SMU (v versus i), `setauto` places both voltage and current ranges in autorange mode.

Example

double icer1;

```
double idatvg[25];
.
.
rangei(SMU1, 2.0E-9);/* Select manual range. */
delay(200);/* Delay after range change. */
measi(SMU1, &icer1);/* Measure leakage. */
.
.
setauto(SMU1);/* Enable autorange mode. */
```

```
                lorangei(SMU1, 2.0E-6);/* Select 2µA as minimum range */
                /* during autoranging. */
                delay(200);/* Delay after range change. */
                smeasi(SMU1, idatvg);/* Setup sweep measurement */
                /* of IDS. */
                sweepv(SMU2, 0.0, 2.5, 24, 0.002);/* Sweep gate from 0 to 2.5 V.
        */
```

setmode Set component mode

Purpose

Set instrument-specific operating mode parameters.

Format

```
int setmode(int instr_id, long modifier, double value);
```

`instr_id`	Instrument ID of the instrument being operated on.
`modifier`	Instrument specific operating characteristic to change. See Table 8-16.
`value`	Value to set the operating parameter to.

Remarks

`Setmode` allows control over certain instrument-specific operating characteristics. Refer to the appendix for the specific instrument for more information on what each instrument supports.

A special instrument ID called KI_SYSTEM is used to set operating characteristics of the system.

For modifier values, refer to Table 8-16, beginning on the next page.

Table 8-16
Modifiers

Support	Parameters			Comment
	instr_id	*modifier*	*value*	
Supported	KI_SYSTEM	KI_TRIGMODE	KI_MEASX KI_INTEGRATE KI_AVERAGE KI_ABSOLUTE KI_NORMAL	Redefines all existing triggers to use a new method of measurement.
		KI_AVGNUMBER	<value>	Number of readings to take when KI_TRIGMODE is set to KI_AVERAGE.
		KI_AVGTIME	<value> (in units of seconds)	Time between readings when KI_TRIGMODE is set to KI_AVERAGE.
No-Op (Accepted but not responded to)[1]		KI_MX_DEFMODE	KI_HIGH KI_LOW	Sets the default matrix mode to high current mode or low current mode. This setting will remain in effect until the end of the current session and is not reset by **devint**.
		KI_HICURRENT	KI_ON	Forces the matrix into high current mode. The mode will revert to the default at the next **devint** unless the configuration file sets this parameter to reset on a **clrcon**.
		KI_CC_AUTO	KI_ON KI_OFF	Turns automatic compliance clear processing on or off. **devint** will reset this value to KI_ON.
		KI_CC_SRC_DLY	<value>	The minimum time after a source value change before a compliance clear scan may start. This represents the time after a source value change it takes the circuit under test to settle and prevent false compliance detection due to transients.
		KI_CC_COMP_DLY	<value>	The time between compliance scans while processing **compclr**. This also represents the time after a source value change it takes the circuit under test to settle and prevent false compliance detection due to transients, but the source value changes are only due to removing instrument from an artificial compliance state.
		KI_CC_MEAS_DLY	<value>	The minimum time after the last source value change before a measurement can be made. This represents the time it takes the circuit under test to settle to the level desired for the subsequent measurements.
Supported	SMUn	KI_INTGPLC	<value> (in units of line cycles)	Specifies the integration time the SMU will use for the **intgx** and **sintgx** commands. The default **devint** value is 1.0. The valid range is 0.01 to 10.0.
		KI_AVGMODE	KI_MEASX KI_INTEGRATE	Controls what kind of readings are taken for **avgX** calls. The **devint** default value is KI_MEASX. When KI_INTEGRATE is specified, the integration time used is that specified by the KI_INTGPLC **setmode** call.

Table 8-16 (continued)
Modifiers

Support	Parameters			Comment
	instr_id	*modifier*	*value*	
	SMUn (continued)	KI_DELAY_FACTOR	<value>	This factor scales the internal delay times used by the SMU. A value larger than one increases the delays; a value less than one decreases the delays. A minimum delay will be enforced by the SMU. NOTE: This command should not be used when setting the SMU speed to FAST, NORMAL, or QUIET modes; the delay factor is set internally by these modes so changing the value while using one of the predefined modes corrupts the speed settings or the delay factor.
No-Op (Accepted but not responded to)[1]	SMUn (continued)	KI_IMTR		Sets up the SMU as a current meter. The ranges used are representative of the type of instrument being simulated. NOTE: This **setmode** will turn the source on.
			KI_S400	Sets the SMU to use ranges equivalent to the Model S400.
			KI_DMM	Sets the SMU to use ranges equivalent to a DMM (lowest range = 100 µa). Provides a lower resolution, fast measurement. Used for high current applications.
			KI_ELECTROMETER	Sets the SMU to use ranges equivalent to an electrometer. Provides best measurement resolution, but has a slower measurement time. Used for low current measurements.
		KI_LIM_INDCTR	Any	Controls what measure value is returned if the SMU is at its programmed limit. The **devint** default is SOURCE_LIMIT (7.0 e22). NOTE: The SMU always returns INST_OVERRANGE (1.0 e22) if it is on a fixed range that is too low for the signal being measured.
		KI_LIM_MODE	KI_INDICATOR KI_VALUE	Controls whether SMU will return an indicator value when in limit or overrange, or the actual value measured. The default mode after a **devint** is to return an indicator value.
No-Op (Accepted but not responded to)[1]	SMUn (continued)	KI_VMTR		Sets up the SMU as a volt meter. The ranges used are representative of the type of instrument being simulated. Note: This **setmode** will turn the source on.
			KI_S400	Sets the SMU to use ranges equivalent to the Model S400.
			KI_DMM	Sets the SMU to use ranges equivalent to a DMM. Provides a low impedance, fast measurement. Used for low voltage applications.
			KI_ELECTROMETER	Sets the SMU to use ranges equivalent to an electrometer. Provides a high input impedance, but has a slower measurement time. Used for high resistance measurements.

1. These modifiers perform no operations in the Model 4200-SCS. They are included only for compatibility, so that existing S600 programs using the setmode function can be ported to the Model 4200-SCS without problems.

sintgX Sweep Integrate

Purpose **sintgX** performs an integrated measurement for every point in a sweep.

Format

```
int sintgi(int instr_id, double *results);
int sintgv(int instr_id, double *results);
```

`inst_id` The measuring instrument's identification code.

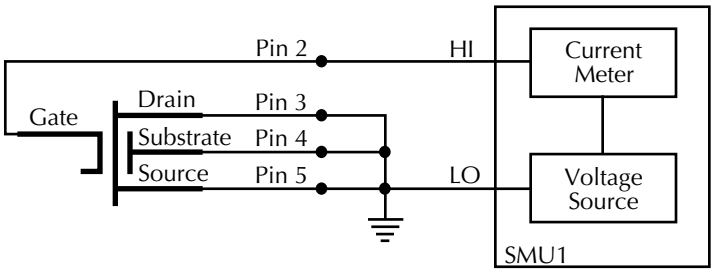
`results` The floating point array where the results are stored.

Remarks This function is used to create an entry in the measurement scan table. During any of the sweeping functions, a measurement scan is performed for every force point in the sweep. During each scan, a measurement will be made for every entry in the scan table. The measurements are made in the same order which the entries were made in the scan table.

`sintgX` sets up the new scan table entry to perform an integrated measurement. The measurement results will be stored in the array specified by results. Each time a measurement scan is made, a new measurement result will be stored at the next location in the results array. If the scan table is not cleared, performing multiple sweeps will simply keep adding new measurement results to the end of the array. Care must be taken to be sure the results array is large enough to hold all measurements which will be taken before the scan table is cleared. The scan table is cleared by an explicit call to `clrscn`, or implicitly when `devint` or `execut` is called.

Example The following example collects information on the low-level gate leakage current of a MOSFET. Sixteen integrated measurements are made as the voltage is increased from 0 to 25.0 V.

Figure 8-106
sintgX




```
double idss [16];
.
.
conpin(SMU1, 2, 0);
conpin(GND, 5, 4, 3, 0);
limiti(SMU1, 1.5E-8);
rangei(SMU1, 2.0E-8);/* Select range for 20nA. */
sintgi(SMU1, idss);/* Measure current with SMU1;*/
/* return results to idss. */
.
.
sweepv(SMU1, 0.0, 25.0, 15, /* Perform 16 measurements */
 1.0E-3);/* (steps) from 0 through */
```

```
./* 25 V; each step 1ms in */
./* duration. */
```

smeasX Sweep measure

Purpose

`smeasX` allows a number of measurements to be made by a specified instrument during a `sweepX` function. The results of the measurements are stored in the defined array.

Format

```
int smeasi(int instr_id, double *results);
int smeast(int instr_id, double *results);
int smeasv(int instr_id, double *results);
```

`instr_id` The measuring instrument's identification code.

`results` The floating point array that stores the results.

Remarks

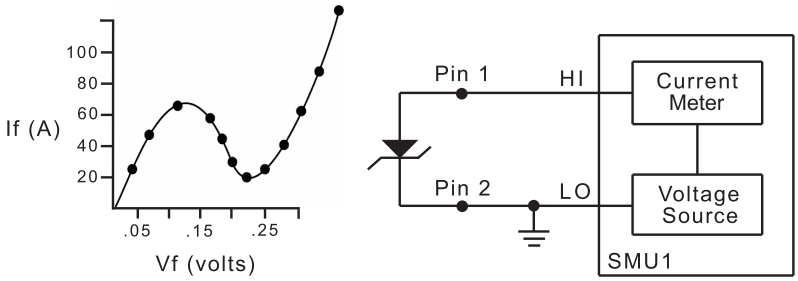
This function is used to create an entry in the measurement scan table. During any of the sweeping functions, a measurement scan is performed for every force point in the sweep. During each scan, a measurement will be made for every entry in the scan table. The measurements are made in the same order in which the entries were made in the scan table.

`smeasX` sets up the new scan table entry to perform an ordinary measurement. The measurement results will be stored in the array specified by results. Each time a measurement scan is made, a new measurement result will be stored at the next location in the results array. If the scan table is not cleared, performing multiple sweeps will simply keep adding new measurement results to the end of the array. Care must be taken to be sure the results array is large enough to hold all measurements that will be taken before the scan table is cleared. The scan table is cleared by an explicit call to `clrscn`, or implicitly when `devint` or `execut` is called.

Example

This example determines the measurement data needed to construct a graph showing the negative resistance characteristics of a tunnel diode. SMU1 generates a voltage ramp ranging from 0 through 0.3 V. The current through the diode is sampled 13 times with a duration of 25 ms at each step. The results are stored in an array called `resi`.

Figure 8-107
smeasX




```
double resi[13];/* Defines array. */
double vf [13];
.
.
.
conpin(SMU1, 1, 0);
conpin(GND, 2, 0);
rtfary (vf);/* Return the voltage force array*/
```

```
smeasi(SMU1, resi);/* Make a series of */
/* measurements; */
./* return the results to the */
./* resi array. */
sweepv(SMU1, 0.0, 0.3, 12,
 25.0E-3);/* Make 13 measurements as the */
/* voltage ranges from 0 to */
/* 0.3V. */
```

sweepX Sweep

Purpose

Generates a ramp consisting of ascending or descending voltages or currents. The sweep consists of a sequence of steps, each with a user-specified duration.

Format

```
int sweepi(int inst_id, double startval, double endval, long
stepno, double step_delay);
```

```
int sweepv(int inst_id, double startval, double endval, long
stepno, double step_delay);
```

`inst_id`	The sourcing instrument's identification code.
`startval`	The initial voltage or current level output from the sourcing instrument and applied for the first sweep measurement. This value can be positive or negative.
`endval`	The final voltage or current level applied in the last step of the sweep. This value can be positive or negative.
`stepno`	The number of current or voltage changes in the sweep. The actual number of forced data points is one greater than the number of steps specified.
`step_delay`	The delay in seconds between each step and the measurements defined by the active measure list.

Remarks

`sweepX` is always used in conjunction with `smeasX`, `sintgX`, `savgX`, or `rtfary`.

`sweepX` causes a sourcing instrument to generate a series of ascending or descending voltages or current changes called steps. During this source time, a measurement scan is performed at each step. The actual number of forced data points is ONE MORE than the number of steps. Thus, the number of measurements performed is the number of steps plus one. This is important when dimensioning the size of the results array. Failure to make sure the array is big enough will produce operating system access violation errors.

Measurements are stored in a one-dimensional array in the order they were taken.

`trigXg`, `trigXl`, and `trigcomp` can be used with `sweepX`, even though they are being used in conjunction with `smeasX`, `sintgX`, or `savgX`. In this case, data resulting from each of the steps is stored in an array, as noted above. However, once a trigger point (for example, a level of current or voltage) is reached, the sourcing device stops incrementing or decrementing and is held at a steady output level for the remainder of the sweep.

The system maintains a measurement scan table consisting of devices to measure. This table is maintained by calls to `smeasX`, `sintgX`, or `savgX`, or `clrscn`. As multiple calls to these functions are made, the commands are appended to this table.

When multiple calls to `sweepX` are executed in the same test sequence, the `smeasX`, `sintgX`, or `savgX` arrays are loaded sequentially. This appends the measurements

from the second `sweepX` call to the previous results. If the arrays are not dimensioned correctly, access violations will occur. The measurement table remains intact until `clrscn`, `devint`, or `execut` are executed.

Defining new test sequences using `smeasX`, `sintgX`, or `savgX` adds commands to the active measure list. The previous measures are still defined and used. `clrscn` is used to eliminate the previous measures for the second sweep. Using `smeasX`, `sintgX`, or `savgX` after `clrscn` causes the appropriate new measures to be defined and used.

In cases where the first sweep point is non-zero, it may be necessary to precharge the circuit so that sweep will return a stable value for the first measured point without penalizing remaining points in the sweep. For example:

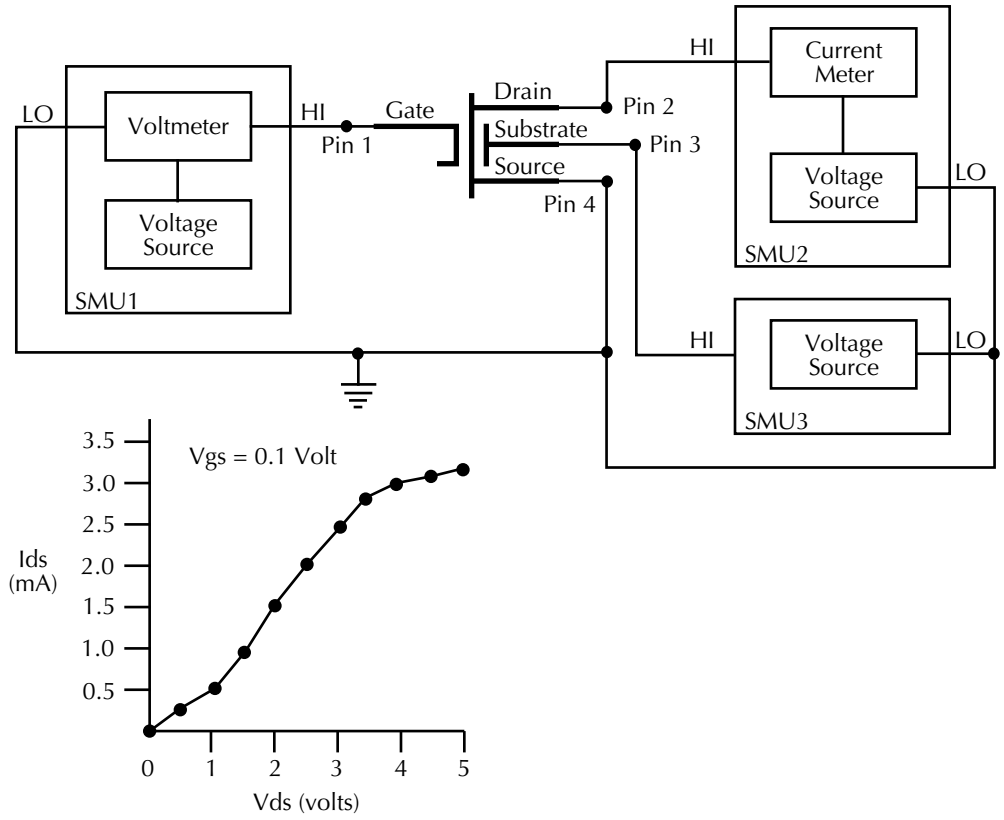
```
double ires[6];
conpin(SMU1, 10, 0);
conpin(2, GND 0);
forcev(SMU1, 5.0);/* Force 5 V to charge. */
delay(10);/* Wait for precharge. */
smeasi(SMU1, ires);/* Set up measurement. */
sweepv(SMU1, 5.0, 10.0, 5, 2.5E-3);/* Do the real measurement.
*/
```

Note that changing the source mode of the SMU can modify the measure range. If the sourcing mode is changed from voltage to current sourcing (or from current to voltage sourcing), the measure range may be changed to minimize variations in the SMU output level. See Changing source mode may change measure range for recommended command order.

Example

The following example gathers data to construct a graph showing the common drain-source characteristics of an FET. A fixed gate-to-source voltage is generated by SMU1. A voltage ramp from 0 through 5 V is generated by SMU2. Drain current applied by SMU2 is measured 11 times by `smeasi`. Data is stored in the array `resi`.

Figure 8-108
sweepX




```
double resi[11], ssbiasv;
double vds[11];
.
conpin(SMU1, 1, 0);
conpin(SMU2, 2, 0);
conpin(SMU3, 3, 0);
conpin(GND, 4, 0);
forcev(SMU3, ssbiasv);/* Apply substrate bias vol- */
/* tage SSBIASV. */
forcev(SMU1, -.1);/* Apply a gate-to-source */
/* voltage of -0.1V. */
rtfary(vds);/* Return force array*/
smeasi(SMU2, resi);/* Perform a series of current */
/* measurements; return */
/* the results to the array */
/* resi. */
sweepv(SMU2, 0.0, 5.0, 10, /* Generate 11 steps and 11 */
 2.5E-3);/* points each 2.5 ms duration, */
/* ranging from 0 to 5 V. */
```

trigcomp Trigger on compliance

Purpose This function will cause a trigger when an instrument goes in or out of compliance.

Format

```
int trigcomp(int instr_id, int mode);
```

`instr_id` The ID of the instrument the trigger is set to.

`mode` Specifies whether to trigger when an instrument is in or out of compliance.
Use 1 to trigger when in compliance.
Use 0 to trigger when out of compliance.

Remarks This function will cause LPT to monitor the given instrument for compliance. A trigger can be set when the instrument is either in compliance or out of compliance, based on the specified mode.

trigXg, trigXl trigXg: trigger if greater than; trigXl: trigger if less than

Purpose Monitors for a predetermined level of voltage, current, or time.

Format

```
int trigig(int inst_id, double value);
int trigil(int inst_id, double value);
int trigtg(int inst_id, double value);
int trigtl(int inst_id, double value);
int trigvg(int inst_id, double value);
int trigvl(int inst_id, double value);
```

`inst_id` The monitoring instrument's identification code.

`value` The voltage, current, or time specified as the trigger point. This trigger point value is considered to be reached when:

- The measured value is equal to or greater than the value argument of `trigXg`, or
- The measured value is less than the value argument of `trigXl`.

Remarks `trigXl` and `trigXg` are used with `searchX` or with one of the `sweep` measurement routines: `smeasX`, `sintgX`, or `savgX`.

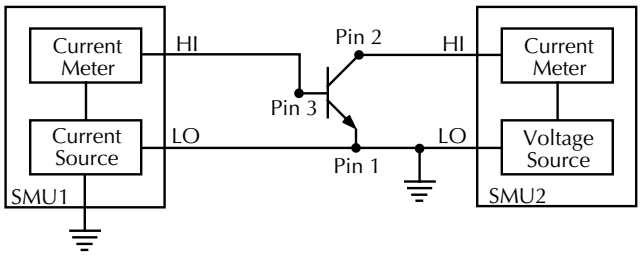
- `trigXg` or `trigXl` provides `sweepX` the digital feedback to allow for the increase or decrease in sourcing values.
- `trigXl` and `trigXg` must be located before any associated `searchX`.
- Triggers are not automatically reset by `searchX` or `sweepX`. A single `trigXl` or `trigXg` can be followed by two or more `searchX` or `sweepX` calls.

The specified trigger point is automatically cleared when a `clrtrg`, `devint`, or `execut` is executed.

Example `trigig` programming example

The following example uses `trigig` and `searchi` together to generate and search for a specific current level. A search is initiated to find the base current needed to produce 5 mA of collector current. The collector-to-emitter voltage supplied by SMU2 is defined by the variable `VCC8`. `searchi` generates the base current from SMU1. This current ranges between 50 μA and 200 μA in 15 iterations. `trigig` continuously monitors the current through SMU1. The base current supplied by SMU1 is stored as the result `res22`.

Figure 8-109
trigXg, trgX1




```
double res22, vcc8;
.
.
conpin(SMU1, 3, 0);
conpin(SMU2, 2, 0);
conpin(GND, 1, 0);
forcev(SMU2, vcc8);/* Apply collector voltage to vcc8. */
trigig(SMU2, +5.0E-3);/* Search for a collector cur- */
/* rent of 5 mA. */
searchi(SMU1, 5.0E-5, 2.0E-4,
 15, 1.0E-3, &res22);/* Generate a current ranging */
/* from 50 to 200µA in */
/* 15 iterations. Return the */
/* current resulting from the */
/* last iteration as res22. */
```

Example `trigil` programming example

The following example uses `trigil` together with `smeasi` and `sweepv` to generate a voltage staircase-type waveform and then measure the resulting current. The waveform holds at its last value when the `trigil` value is reached. The data needed to generate a graph showing the forward voltage-to-current characteristics of a diode are stored in an array. SMU1 is configured as a voltage source and current meter. The `sweepv` sources voltage from SMU1. This voltage staircase ranges from 0.6 V to 0.0 V in 18 steps with a 1 ms duration at each step. `trigil` stops the ramping initiated by `sweepv` when the current through the diode is greater than +4 mA. `smeasi` measures the current applied by SMU1 at each voltage step and stores the results in array `res1`.

```
double res1[20];
.
.
conpin(SMU1, 1, 0);
conpin(GND,2, 0);
trigil(SMU1, +4.0E-3);/* If less than +4mA, */
/* stop ramping. */

smeasi(SMU1, res1);/* Measure current at each of */
/* the 19 levels; return re- */
/* sults to the res1 array. */
sweepv(SMU1, 0.0, 0.6, 18,
 1.00E-3);/* Generate 0.0 to 0.6V */
/* in 18 steps. */
```

tstdsl Test station deselect

Purpose Used to deselect a test station.

Format
```
tstdsl (void);
```

Remarks To relinquish control of an individual test station, a new test station must now be selected using `tstsel` before any subsequent test control functions are run.

The `tstdsl` command has the same effect as the `tstsel (0)` command.

NOTE `tstdsl` *is not required for use in a UTM.*

Example
```
tstdsl ( );  /* Disables test station.*/
```

See also `tstsel`

tstsel Test station select

Purpose Used to enable or disable a test station.

Format
```
tstsel (int x);
```

Where `x` is the test station number, 0 or 1.

Remarks `tstsel` is normally called at the beginning of a test program.

`tstsel (1)` will select the first test station and load the instrumentation configuration.

NOTE `tstsel` *is not required for use in a UTM.*

See also `tstdsl`

LPT functions for the Model 4205-PG2

The following information explains the functions included in the Keithley LPTLib (Linear Parametric Test Library) for the Model 4205-PG2 pulse generator card. The functions are summarized in Table 8-11.

NOTE *All pulse functions are supported by the Model 4205-PG2 pulse generator card. Most pulse functions are also supported by the Model 4200-PG2 if the firmware is upgraded to KITE 6.2. Instructions for upgrading firmware are available by clicking on the Complete Reference icon on the Model 4200-SCS desktop. Follow the links for Release Notes, then look for the Firmware Upgrade Procedure for the pulse card.firmware.*

NOTE *The terms "pulse generator card" and "PG2" will be used for functions that pertain to both the Model 4205-PG2 and Model 4200-PG2. Operations that are not supported by the Model 4200-PG2 are explained.*

arb_array Defines a full-arb waveform

Purpose This function is used to define a full-arb waveform and name the file.

Format

```
int arb_array(INSTR_ID instr_id, long ch, double TimePerPt, long
length, double *levelArr, char *fname);
```

instr_id	Instrument ID of the PG2: VPU1, VPU2, and so on.
`ch`	The PG2 channel: 1 or 2.
`TimePerPt`	Sets the time interval between waveform points: 20 ns to 1 s.
`length`	The number of points (values): 262,144 maximum.
`levelArr`	An array of voltage values for each point in the waveform.
`fname`	A name for the full-arb waveform.

Remarks This function is used to define the number of points in a waveform, the time interval between points, and the voltage value at each point. The maximum number of waveform points per channel is 262,144.

The load time for a full-arb waveform is proportional to the number of points. The total time to load full-size full-arb waveforms for both channels is around one minute.

Once loaded, use pulse_output to turn on the appropriate channels, and then use pulse_trig to select the trigger mode and start (or arm) pulse output.

Refer to Full Arb in Section 11 for details on this pulse mode. The following voltage level array is required for the example full-arb waveform shown in Table 8-17.

Table 8-17
arb_array

Level array	Level array (continued)
levelArr(0) = 0.5	levelArr(41) = 19.5
levelArr(1) = 1.0	levelArr(42) = 19.0
levelArr(2) = 1.5	levelArr(43) = 18.5
•	•
•	•
•	•
levelArr(39) = 19.5	levelArr(79) = 0.5

.kaf waveform file for KPulse: The arbitrary waveform data defined by the arb_file function can be copied into a `.kaf` file. Use a text editor to properly format the file. The `.kaf` file can then be imported into KPulse. By default, `.kaf` waveform files for KPulse are saved in the `ArbFiles` folder at the following command path location: `C:\S4200\kiuser\KPulse\ArbFiles`. Refer to Section 13 for details on using KPulse.

See also **seg_arb_define**

arb_file Loads a waveform from a full-arb waveform file

Purpose This function is used to load a waveform from an existing full-arb waveform file.

Format

```
int arb_file(INSTR_ID instr_id, long ch, char *fname)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`ch`	The PG2 channel: 1 or 2.
`fname`	The name of the waveform file.

Remarks This function is used to load a waveform from an existing full-arb `.kaf` waveform file into the pulse generator card. A full-arb waveform can be loaded for each channel of the pulse generator card. Once loaded, use pulse_output to turn on the appropriate channel, and then use pulse_trig to select the trigger mode and start (or arm) pulse output.

When specifying the `fname`, include the full command path with the file name. Existing `.kaf` waveforms are typically saved in the `ArbFiles` folder at the following command path location:

```
C:\S4200\kiuser\KPulse\ArbFiles
```

A full-arb waveform can be created using KPulse, and then saved as a `.kaf` waveform file (refer to Section 13 for details).

A waveform in an existing `.kaf` file can be modified in two ways:

- Use a text editor to modify.
- Import into KPulse and then modify.

See also **arb_array**, **seg_arb_file**, **seg_arb_define**

Example The following function loads a full-arb file named `SINE.kaf` (saved in the `ArbFiles` folder) into the pulse generator card for Channel 1:

```
arb_file(VPU1, 1,
"C:\\S4200\\kiuser\\KPulse\\ArbFiles\\SINE.kaf")
```

pg2_init Resets PG2 to default settings for specified pulse mode

Purpose Use this function to change the pulse mode. it resets the pulse generator card to the specified pulse mode (standard, full-arb, or Segment ARB) and its default conditions

Table 8-18
pg2_init

Standard pulse defaults	Full Arb and Segment ARB pulse defaults
Pulse high and pulse low = 0 V	Source range = 5 V; fast speed
Source range = 5 V; fast speed	Pulse count = 1
Pulse period = 1 µs	Pulse delay = 0 s
Pulse width = 500 ns	Pulse load = 50 Ω
Pulse count = 1	Pulse trigger source = Software
Rise and fall time = 100 ns	Trigger mode = Continuous
Pulse delay = 0 s	Pulse trigger output = Off*
Pulse load = 50 Ω	Trigger polarity = Positive
Pulse trigger source = Software	Current limit = 105 mA
Trigger mode = Continuous	Pulse output = Off
Pulse trigger output = On*	
Trigger polarity = Positive	
Complement mode = Normal Pulse	
Current limit = 105 mA	
Pulse output = Off	

* Turns on when a pulse is initiated with pulse_trig.

Format

```
int pg2_init(INSTR_ID instr_id, long mode)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`mode`	Pulse mode: 0 (standard pulse), 1 (Segment Arb) or 2 (full-arb).

Remarks This function resets both channels of the PG2 to the default settings of the specified pulse mode. The default setting for each parameter is listed in the Format section for each LPT function.

The pulse_init function can be used to reset the pulse generator card to the default settings for the presently selected pulse mode.

Example The following function resets the PG2 to the Segment ARB pulse mode and its default settings:

```
pg2_init(VPU1, 1)
```

pulse_burst_count Sets count for pulse burst mode

Purpose For the burst mode, this function sets the number of pulses to output during a burst sequence.

Format

```
int pulse_burst_count(INSTR_ID instr_id, long chan, unsigned long count)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`count`	Number of pulses to output: 1 to (2^{32}-1). Default: `1`

Remarks

Each channel of the PG2 can have a unique burst count. When a burst sequence is triggered, the PG2 will output the specified number of pulses and then stop. The pulse_trig function is used to start (or arm) the burst sequence (`Burst` or `Trig Burst`).

NOTE With an external trigger source selected, the burst count for channel 1 cannot be less than the burst count for channel 2. Setting the burst count for channel 2 higher than the burst count for channel 1 may cause your system to stop responding when pulse output is triggered to start. Also, when using one channel, set the unused channel to the same burst count value. See pulse_trig_source for details on selecting an external trigger source.

Example

The following function sets the burst count for the PG2 channel 1 to 10:

```
pulse_burst_count(VPU1, 1, 10)
```

pulse_current_limit Sets current limit for the PG2

Purpose

This function sets the current limit of the PG2.

Format

```
int pulse_current_limit(INSTR_ID instr_id, long chan, double ilimit)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`ilimit`	Current limit value (in amps, range and load dependent): 5 V range: -0.2 to +0.2 20 V range: -0.8 to +0.8 Default: `0.105 (5 V range)`

Remarks

Current limit can be independently set for each PG2 channel. Current limit values are range dependent and can be set from -0.2 A to +0.2 A (5 V range, 50 Ω load) or -0.4 A to +0.4 A (20 V range, 50 Ω load). Current limit is used to protect the DUT by using the specified DUT load to calculate the voltage required to reach the current limit. A PG2 channel will not exceed the voltage required to reach the set current limit value at the specified DUT load.

See also

`pulse_load`

Example

The following function sets the current limit of PG2 channel 1 to 1 mA:

```
pulse_current_limit(VPU1, 1, 1e-3)
```

pulse_dc_output Selects DC output and sets voltage level

CAUTION **The** `pulse_vlow`**,** `pulse_vhigh`**, and** `pulse_dc_output` **commands set the voltage value output by the pulse channel when it is turned on (using** `pulse_output`**). If the output is already enabled, these commands will change the voltage level immediately, even before the pulsing is started with a** `pulse_trig` **command.**

Purpose This function selects the DC output mode and sets the voltage level.

Format

```
int pulse_dc_output(INSTR_ID instr_id, long chan, double dcvalue)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`dcvalue`	DC voltage output value (in volts, range, and load dependent): 5 V range: -5 to +5 20 V range: -20 to +20 Default: N/A

Remarks Each PG2 channel can be set to output a fixed DC voltage level, rather than pulses. The DC output for each PG2 channel can be set from -5 V to +5 V (5 V range) or -20 V to +20 V (20 V range) for 50 Ω load.

See also **`pulse_load`**

Example The following function selects channel 1 DCV output and sets voltage to +10 V:

```
pulse_dc_output(VPU1, 1, 10)
```

pulse_delay Sets time delay from trigger to pulse output

Purpose This function sets the delay period from trigger to when pulse output starts.

Format

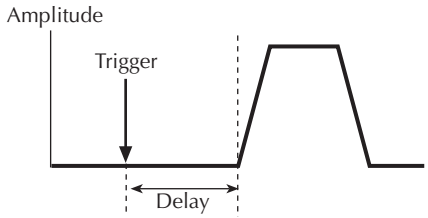
```
int pulse_delay(INSTR_ID instr_id, long chan, double delay)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`delay`	Time delay in seconds: Fast speed: 0 to (Period - 10 e-9) Slow speed: 0 to (Period - 10 e-9) Default: 0

Remarks Pulse delay can be set independently for each PG2 channel. For both speeds, pulse delay can be set from 0 ns to (Period - 10 ns). The pulse_range function is used to set pulse speed.

As shown below, pulse delay is the time from pulse trigger initiation to the start of the rise transition time.

Figure 8-110
pulse_delay



The maximum pulse delay that can be set depends on the presently set period for the pulse. For example, if the period is set for 500 ns, the maximum pulse delay that can be set is 490 ns (500 ns - 10 ns = 490 ns).

NOTE *Use the pulse_source_timing function to set the pulse delay time for the Models 4220-PGU and 4225-PMU.*

See also **pulse_period**, **pulse_trig**

Example The following function sets the pulse delay for channel 1 to 300 ns:

```
pulse_delay(VPU1, 1, 300e-9)
```

pulse_fall Sets pulse fall time

Purpose This function sets the fall transition time for the PG2 pulse output.

Format

```
int pulse_fall(INSTR_ID instr_id, long chan, double fallt)
```

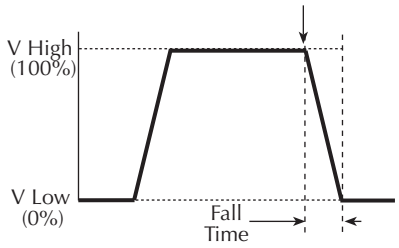
instr_id	Instrument ID of the PG2: VPU1, VPU2, and so on.
chan	Channel number of the PG2: 1 or 2.
fallt	Pulse fall time in seconds (floating point number) Fast speed: 10 e-9 to 33 e-3 Slow speed[8]: 50 e-9 to 33 e-3 Default: 100 e-9

Remarks Rise and fall transition time can be set independently for each PG2 channel. There is a minimum slew rate for both the rise and fall transitions. For the fast speed range, the minimum is 362 µV/µs, or 1 V/2.7 ms. For the high voltage range, the minimum slew rate is 1.8 mV/µs, or 1 V/500 µs.The pulse_range function is used to set pulse speed.

As shown below, the pulse fall time occurs between the 100 percent and 0 percent amplitude points on the falling edge of the pulse, where the amplitude is the difference between the V High and V Low pulse values.

8. Note that the minimum transition time for pulse source only (no measurement) on the 40 V range is 50 ns for the Model 4225-PMU and Model 4220-PGU. The Model 4200-PG2 and Model 4205-PG2 have a minimum transition time of 100 ns.

Figure 8-111
pulse_fall



The pulse fall time setting takes effect immediately during continuous pulse output. Otherwise, the fall time setting takes effect when the next trigger is initiated. The `pulse_trig` function is used to trigger continuous or burst output.

NOTE *Use the* `pulse_source_timing` *function to set the pulse fall time for the Models 4220-PGU and 4225-PMU.*

See also

Example For fast speed, the following function sets the pulse fall time for channel 1 of the PG2 to 50 ns:

```
pulse_fall(VPU1, 1, 50e-9)
```

pulse_halt Stops pulse output

Purpose This function stops all pulse output from the PG2.

Format

```
int pulse_halt(INSTR_ID instr_id)
```

`instr_id` Instrument ID of the PG2: VPU1, VPU2, and so on.

Remarks This function stops all pulse output from the PG2 and turns the PG2 channels off. Pulse output can be restarted by first turning the outputs back on with `pulse_output` and then using the `pulse_trig` function to restart the test.

See also

Example The following function stops PG2 pulse output:

```
pulse_halt(VPU1)
```

pulse_init Resets PG2 to default settings for the present pulse mode

Purpose This function resets the PG2 to the default settings for whichever pulse mode (standard, full-arb, or Segment ARB) is presently selected:

Table 8-19
pulse_init

Standard pulse defaults	Full Arb and Segment ARB pulse defaults
Pulse high and pulse low = 0 V	Source range = 5 V: Fast speed
Source range = 5 V: fast speed	Pulse count = 1
Pulse period = 1 µs	Pulse delay = 0 s
Pulse width = 500 ns	Pulse load = 50 Ω
Pulse count = 1	Pulse trigger source = Software
Rise and Fall time = 100 ns	Pulse trigger mode = Continuous
Pulse delay = 0 s	Pulse trigger output = Off
Pulse load = 50 Ω	Trigger polarity = Positive
Pulse trigger source = Software	Current limit = 105 mA
Pulse trigger mode = Continuous	Pulse output = Off
Pulse trigger output = On	
Trigger polarity = Positive	
Complement mode = Normal pulse	
Current limit = 105 mA	
Pulse output = Off	

Format

```
int pulse_init(INSTR_ID instr_id)
```

`instr_id` Instrument ID of the PG2: VPU1, VPU2, and so on.

Remarks This function resets both channels of the PG2 to the default settings. The default setting for each parameter is listed in the Format section for each LPT function.

The `pg2_init` function can be used to specify which pulse mode to reset. Using this function selects the specified pulse mode and its default settings.

See also **pg2_init**

Example The following function resets the PG2 to the default settings for the presently selected pulse mode:

```
pulse_init(VPU1)
```

pulse_load Sets output impedance for the load

Purpose This function sets the output impedance for the load (DUT).

Format

```
int pulse_load(INSTR_ID instr_id, long chan, double load)
```

`instr_id` Instrument ID of the PG2: VPU1, VPU2, and so on.

`chan` Channel number of the PG2: 1 or 2.

`load` Output impedance (in ohms): 1 to 10 e6

Default: 50

Remarks DUT impedance can be independently set for each PG2 channel. The DUT impedance can be set from 1 Ω to 10 M Ω, depending on the programmed pulse high and low values.

Maximum power transfer is achieved when the DUT impedance matches the output impedance of the PG2. For example, if the DUT impedance is set to 1 MΩ, the voltage output settings will change to account for the higher DUT impedance, ensuring that the voltage at the DUT will not be double the voltage setting (caused by reflection due to load mismatching).

Example The following function sets the output impedance of PG2 channel 1 to 100 Ω:

```
pulse_load(VPU1, 1, 100).
```

pulse_output Sets pulse output on or off

Purpose This function sets the pulse output of a PG2, PGU, or PMU (without RPM) channel on or off. This command also prepares the pulse source when using a PMU with RPM(s). See the first note in this function's remarks section for details.

Format

```
int pulse_output(INSTR_ID instr_id, long chan, long out_state)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`out_state`	Pulse output state: 0 (off) or 1 (on). Default: 0 (off)

Remarks When a PG2 channel is off, the output is in a high-impedance (open) state. The command configures the channel to output and close the output relay. This connects the source to the device under test (DUT). When using the PG2 commands (see Table 8-16 on page 8-144), the pulse channel starts pulsing when a trigger is initiated. Note that if a pulse delay has been set, pulse output will start after the delay period expires. Also, `devclr()` resets the VPU source and disconnects the source from the DUT.

NOTE *When using a Model 4225-RPM with a PMU, the pulse_output command is necessary to prepare the channel for sourcing, but does not close the output relay. Use the rpm_config command to enable the RPM output.*

This function does not control the high-endurance output relays (HEORs) on the Model 4205-PG2 card. The `pulse_ssrc` function controls (open/close) the HEORs, and the `seg_arb_define` function is used to define a Segment ARB® waveform, which includes HEOR control.

NOTE *It is good practice to routinely turn off the outputs of the PG2 after a test has been completed.*

See also , , **pulse_current_limit**

Example The following function turns off the output for PG2 channel 1:

```
pulse_output(VPU1, 1, 0)
```

pulse_output_mode Sets pulse output mode

Purpose This function sets the pulse output mode of a PG2 channel.

Format

```
int pulse_output_mode(INSTR_ID instr_id, long chan, long mode)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`mode`	Pulse output state: NORMAL (0) or COMPLEMENT (1). Default: NORMAL

Remarks When a PG2 channel is in COMPLEMENT mode, the V Low and V High voltage settings are swapped.

Example The following function sets the output mode for PG2 channel 1 to COMPLEMENT:

```
pulse_output_mode(VPU1, 1, COMPLEMENT)
```

pulse_period Sets pulse period

Purpose This function sets the period for pulse output.

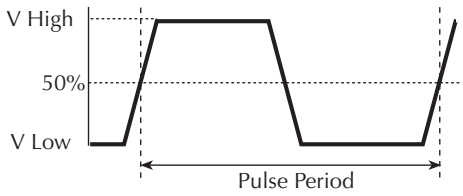
Format

```
int pulse_period(INSTR_ID instr_id, double period)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`period`	Pulse period (in seconds): 5 V range: 20 e-9 to 1 20 V range: 500 e-9 to 1 Default: 1 e-6

Remarks This function sets the pulse period for both channels of the PG2. As shown below, the pulse period is measured at the median point (50 percent between the high and low pulse values) from the rising edge of a pulse to the rising edge of the next pulse.

Figure 8-112
pulse_period



The pulse period setting takes effect immediately during continuous pulse output. Otherwise, the period setting takes effect when the next trigger is initiated. The function is used to trigger continuous or burst output.

Example The following function sets the pulse period of the PG2 to 200 ns:

```
pulse_period(VPU1, 200e-9)
```

pulse_range Sets pulse voltage range (low or high)

Purpose Sets a PG2 channel for low voltage (fast speed) or high voltage (slow speed).

Format

```
int pulse_range(INSTR_ID instr_id, long chan, double range)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`range`	Pulse range (in volts): 5 or 20. Default: 5 V

Remarks Setting the pulse range of a PG2 channel to 5 V selects the low-voltage range. Selecting the low-voltage range also selects fast speed for pulse output. For fast speed, the minimum pulse width that can be set is 10 ns, and minimum rise / fall times can be set to 10 ns.

Setting the pulse range of a PG2 channel to 20 V selects the high-voltage range. Selecting the high-voltage range also selects slow speed for pulse output. For slow speed, the minimum pulse width that can be set is 250 ns, and the minimum rise / fall times can be set to 100 ns.

This setting takes effect when the next trigger is initiated. The following pulse parameters are then checked: period, width, rise time, fall time, and high and low voltage levels. If any of these parameters is out of bounds, it is reset to the default value.

NOTE *When using the function, changing the source range after setting voltage levels, in any pulse mode, may result in voltage levels invalid for the new range setting. Therefore, use* `pulse_range` *before setting the voltage levels.*

NOTE *This function (*`pulse_range`*) can also be used to set the voltage source range of the Models 4220-PGU and 4225-PMU. Use the function to set the source and measure ranges of the Model 4225-PMU.*

See also **`pulse_fall`**, **`pulse_vhigh`**, **`pulse_vlow`**, **`pulse_period`**, **`pulse_rise`**, **`pulse_width`**

Example The following function selects the high-voltage (slow speed) range for PG2 channel 1:

```
pulse_range(VPU1, 1, 20).
```

pulse_rise Sets pulse rise time

Purpose This function sets the rise transition time for the PG2 pulse output.

Format

```
int pulse_rise(INSTR_ID instr_id, long chan, double riset)
```

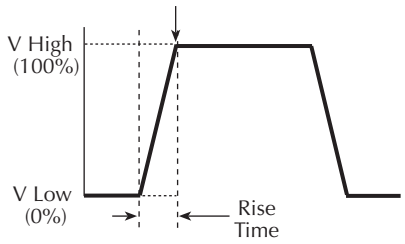
`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`riset`	Pulse rise time in seconds (floating point number) Fast speed: 10 e-9 to 33 e-3 Slow speed[9]: 50 e-9 to 33 e-3

Default: 100 e-9

Remarks Rise and fall transition time can be set independently for each PG2 channel. There is a minimum slew rate for both the rise and fall transitions. For the fast speed range, the minimum is 362 µV/µs, or 1 V/2.7 ms. For the high-voltage range, the minimum slew rate is 1.8 mV/µs, or 1 V/500 µs.The function is used to set pulse speed.

As shown below, the pulse rise time is measured from the 0 percent and 100 percent amplitude points on the rising edge of the pulse, where the amplitude is the difference between the V High and V Low values.

Figure 8-113
pulse_rise



The pulse rise time setting takes effect immediately during continuous pulse output. Otherwise, the rise time setting takes effect when the next trigger is initiated. The function is used to trigger continuous or burst output.

NOTE *Use the function to set the pulse rise time for the Models 4220-PGU and 4225-PMU.*

See also `pulse_fall`

Example For fast speed, the following function sets the pulse rise time for channel 1 of the PG2 to 50 ns:

```
pulse_rise(VPU1, 1, 50e-9)
```

pulse_ssrc Control the high endurance output relays on the Model 4205-PG2

Purpose This function controls the high endurance output relay (HEOR) for each output channel of the Model 4205-PG2.

Format

```
int pulse_ssrc(INSTR_ID instr_id, long chan, long state, long ctrl)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`state`	0 (Open) or 1 (Close)
`ctrl`	0 (Auto), 1 (Manual) or 2 (Trigger out driven) Default: 1 (Close), 0 (Auto)

9. Note that the minimum transition time for pulse source only (no measurement) on the 40 V range is 50 ns for the Model 4225-PMU and Model 4220-PGU. The Model 4200-PG2 and Model 4205-PG2 have a minimum transition time of 100 ns.

Remarks When a high-endurance output relay (HEOR) for a Model 4205-PG2 channel is opened, the output for that channel will be electrically isolated from the DUT. Note that this setting is independent of the output relay (see).

The `ctrl` parameter determines how the HEOR will be controlled. When set to 0 (auto), the Segment ARB pulse mode will control the HEOR. When set to 2 (trigger out driven), the relay state will follow the trigger output. When `ctrl` is set to 1 (manual), the `state` parameter will open (0) or close (1) the HEOR.

The Model 4200-PG2 does not have high endurance output relays. Calling the `pulse_ssrc` function for the Model 4200-PG2 will return an error.

Example The following function selects Manual control and opens the HEOR:

```
pulse_ssrc(VPU1, 1, 0, 1)
```

pulse_trig Select trigger mode and initiate or arm pulse output

Purpose This function selects the trigger mode (continuous, burst, or trig burst) and initiates the start of pulse output or arms the pulse generator card.

Format

```
int pulse_trig(INSTR_ID instr_id, long mode)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`mode`	Trigger mode: 0 (burst), 1 (continuous) or 2 (trig burst)

Remarks With the software trigger source selected, this function will set the trigger mode (continuous, burst, or trig burst) for both pulse generator card channels, and initiate the start of pulse output.

Model 4205-PG2 only: With an external trigger source selected, this function will set the trigger mode, and arm the Model 4205-PG2. Pulse output will start when an external trigger is applied to the Trigger In connector. The Model 4200-PG2 does support input triggering (no Trigger In connector).

The function is used to select the trigger source for the Model 4205-PG2.

In the continuous trigger mode, the PG2 will output pulses continuously. For burst and trig burst, the PG2 will output the programmed number of pulses and then stop.

NOTE *See Triggering in Section 11 for details on triggering.*

If pulse delay is set to zero (0), pulse output will start immediately after it is triggered. If pulse delay is >0, pulse output will start after the delay period expires.

See also **`pulse_burst_count`**, **`pulse_delay`**

Example The following function initiates (triggers) burst pulse output:

```
pulse_trig(VPU1, 0)
```

pulse_trig_output Sets output trigger on or off

Purpose This function sets the output trigger on or off.

Format

```
int pulse_trig_output(INSTR_ID instr_id, long state)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`state`	Output trigger state: 0 (off) or 1 (on) Default: 1 for standard pulse, 0 for Segment ARB and full arb

Remarks This function turns the TTL level trigger output pulse on or off. The pulse used to synchronize pulse output with the operations of an external instrument. When connected to a scope, each output pulse of the PG2 will trigger a scope waveform measurement.

See also **`pulse_trig_polarity`**

Example The following function sets the PG2 trigger output on:

```
pulse_trig_output(VPU1, 1)
```

pulse_trig_polarity Sets polarity of the output trigger

Purpose This function sets the polarity (positive or negative) of the PG2 output trigger.

Format

```
int pulse_trig_polarity(INSTR_ID instr_id, long polarity)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`polarity`	Output trigger polarity: 0 (negative) or 1 (positive) Default: 1 (positive, rising edge)

Remarks Trigger output provides a TTL level output that is at the same frequency (period) as the PG2 output channels. It is used to synchronize pulse output with the operations of an external instrument. When connected to a scope, each output pulse of the PG2 will trigger a scope waveform measurement.

The external instrument that is connected to the PG2 external trigger may require a positive-going (rising-edge) pulse or a negative-going (falling-edge) pulse for triggering. If using the scope card, the PG2 output trigger must be set for positive polarity.

If a `polarity` value other than 0 or 1 is sent, it will map to 0 or 1 in the following manner:

```
if(polarity <= 0)
pol = NEGATIVE;
else
pol = POSITIVE;
```

NOTE *Models 4220-PGU and 4225-PMU: Do not use the two external **falling** trigger sources (see function) with the positive trigger output polarity on the master card that triggers itself and other subordinate cards. These two falling trigger sources should only be used when an external piece of equipment is being used to supply the trigger pulses to the Models 4220-PGU and 4225-PMU. This applies to all three pulse modes (standard pulse, Segment ARB and full arb).*

NOTE *When triggering multiple Model 4205-PG2 cards in a master/subordinate configuration, changing the master trigger output polarity (*`pulse_trigger_polarity` *function) will result in a transition in the trigger output levels that may be interpreted as a trigger pulse by the other cards.*

See also `pulse_trig_output`

Example The following function sets the PG2 trigger output for negative polarity:

```
pulse_trig_polarity(VPU1, 0)
```

pulse_trig_source Sets trigger source

Purpose This function sets the trigger source

Format

```
int pulse_trig_source(INSTR_ID instr_id, long source)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`source`	Trigger source: 0 (software) 1 (external – initial trigger only – rising) 2 (external – initial trigger only – falling) 3 (external – trigger per pulse – rising) 4 (external – trigger per pulse – falling) 5 (internal trigger bus) Default: Software

Remarks This function sets the trigger source that will be used to trigger the Model 4205-PG2 to start its output. With the software trigger source selected, the function will select the trigger mode (continuous, burst, or trig burst), and initiate the start of pulse output.

NOTE *Models 4220-PGU and 4225-PMU: Do not use the two external* ***falling*** *trigger sources with the positive trigger output polarity (see* `pulse_trig_polarity` *function) on the master card that triggers itself and other subordinate cards. These two falling trigger sources should only be used when an external piece of equipment is being used to supply the trigger pulses to the Models 4220-PGU and 4225-PMU. This applies to all three pulse modes (standard pulse, Segment ARB and full arb).*

NOTE *The Model 4200-PG2 does not support input triggering. Calling the* `pulse_trig_source` *function for the Model 4200-PG2 will return an error.*

NOTE *Because trigger source is really a card-level setting and NOT a channel setting, using channel 1 or 2 will set the card to the specified source card 1. Similarly, channel 3 or 4 will set the source for card 2.*

With an external trigger source selected, the `pulse_trig` function will select the trigger mode and arm pulse output. Pulse output will start when the required external trigger pulse is applied to the Trigger In connector. There is a trigger-in delay of 560 ns. This is the delay from the trigger-in pulse to the time of the rising edge of the output pulse.

For an initial trigger only setting, only the first rising or falling trigger pulse will start and control pulse output.

For a trigger per pulse setting, rising or falling edge trigger pulses will start and control pulse output. After the initial pulse, the pulse output, either continuous or burst, will be output based on the internal pulse generator clock. If pulse-to-pulse synchronization is required over higher count pulse trains, use the trigger per pulse mode. For details, refer to External triggering in Section 11.

The internal bus trigger source is used for synchronizing multiple PMU/PGU cards for standard pulse using the legacy pulse functions (`pulse_vhigh`, `pulse_vlow`, `pulse_width`, and so on). This trigger source is used only by the Models 4220-PGU and 4225-PMU.

See also **`pulse_trig`**

Example The following function sets the trigger source to external – initial trigger only – rising:

```
pulse_trig_source(VPU1, 1)
```

pulse_vhigh Sets pulse V High value

Purpose This function sets the pulse V High level.

Format

```
int pulse_vhigh(INSTR_ID instr_id, long chan, double vhigh)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`vhigh`	Pulse V High value in volts (floating point number) Fast speed: -5 to +5 Slow speed: -20 to +20 Default: 0

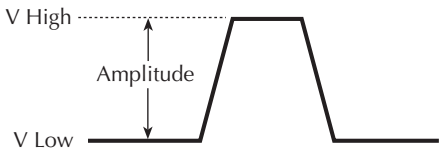
Remarks Pulse V High can be set independently for each pulse card channel. For lower voltages and faster transitions, it can be set from -5 V to 5 V for 50 Ω DUT load. For higher voltages and slower transitions, it can be set from -20 V to 20 V for 50 Ω DUT load. The `pulse_range` function sets the pulse voltage range.

NOTE *When using the* `pulse_range` *function, changing the source range after setting voltage levels (in any pulse mode) will result in voltage levels invalid for the new range setting. Therefore, use* `pulse_range` *before setting the voltage levels.*

As shown below, the pulse V High is typically set as the greater pulse voltage value. However, V High can be any valid voltage value. That means pulse V High can be less than V Low. When started, the pulse transitions from V Low to V High and then back to V Low. The voltage remains at V Low for the remainder of the pulse period.

The pulse V High setting takes effect immediately during continuous pulse output. Otherwise, the V High setting takes effect when the next trigger is initiated. The pulse_trig function is used to trigger continuous or burst output.

Figure 8-114
pulse_vhigh



See also `pulse_vlow`

Example The following function sets the pulse V High value for channel 1 of the PG2 to 2.5 V:

```
pulse_vhigh(VPU1, 1, 2.5)
```

pulse_vlow Sets pulse V Low value

CAUTION **The `pulse_vlow` and `pulse_dc_output` commands set the voltage value output by the pulse channel when it is turned on (using `pulse_output`). If the output is already enabled, these commands will change the voltage level immediately, even before the pulsing is started with a `pulse_trig` command.**

Purpose

This function sets the pulse V Low value.

Format

```
int pulse_vlow(INSTR_ID instr_id, long chan, double vlow)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`vlow`	Pulse V Low value in volts (floating point number) Fast speed: -5 to +5 Slow speed: -20 to +20 Default: 0

Remarks

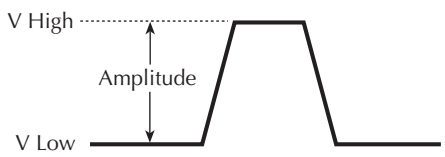
Pulse V Low can be set independently for each pulse card channel. For lower voltages and faster transitions, it can be set from -5 V to 5 V for 50 Ω DUT load. For higher voltages and slower transitions, it can be set from -20 V to 20 V for 50 Ω DUT load. The `pulse_range` function sets the pulse voltage range.

NOTE *When using the `pulse_range` function, changing the source range after setting voltage levels, in any pulse mode, will result in voltage levels invalid for the new range setting. Therefore, use `pulse_range` before setting the voltage levels.*

As shown below, the pulse V Low is typically set as the lesser pulse voltage value. However, V Low can be any valid voltage value. That means pulse V Low can be higher than V High. When started, the pulse transitions from V Low to V High and then back to V Low. The voltage remains at V Low for the remainder of the pulse period.

The pulse V Low setting takes effect immediately during continuous pulse output. Otherwise, the V Low setting takes effect when the next trigger is initiated. The pulse_trig function is used to trigger continuous or burst output.

Figure 8-115
pulse_vlow



See also **`pulse_vhigh`**

Example

The following function sets the pulse V Low value for channel 1 of the PG2 to 0.5 V:

```
pulse_vlow(VPU1, 1, 0.5)
```

pulse_width Sets pulse width

Purpose This function sets the pulse width for pulse output.

Format

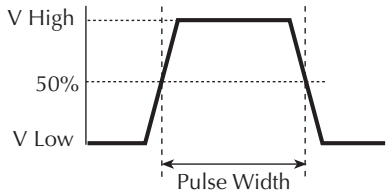
```
int pulse_width(INSTR_ID instr_id, long chan, double width)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`chan`	Channel number of the PG2: 1 or 2.
`width`	Pulse width in seconds Fast speed: 10 e-9 to (Period - 10 e-9) Slow speed: 250 e-9 to (Period - 10 e-9) Default: 500 e-9

Remarks Pulse width can be independently set for each PG2 channel. For fast speed, pulse width can be set from 10 ns to (Period - 10 ns). For slow speed, pulse width can be set from 250 ns to (Period - 10 ns). The `pulse_range` function is used to set pulse speed.

PG2 pulse width is based on the full width, half max method (FWHM). As shown below, the pulse width is measured at the median (50 percent amplitude) point from the rising edge of the pulse to the falling edge of the pulse.

Figure 8-116
pulse_width



The maximum pulse width that can be set depends on the presently set period for the pulse. For example, if the period is set for 500 ns, the maximum pulse width that can be set is 490 ns (500 ns - 10 ns = 490 ns).

The pulse width setting takes effect immediately during continuous pulse output. Otherwise, the width setting takes effect when the next trigger is initiated. The `pulse_trig` function is used to trigger continuous or burst output.

NOTE *Use the* `pulse_source_timing` *function to set the pulse width for the Models 4220-PGU and 4225-PMU.*

See also **`pulse_period`**

Example The following function sets the pulse width for channel 1 to 250 ns:

```
pulse_width(VPU1, 1, 250e-9)
```

seg_arb_define Defines a Segment ARB waveform

Purpose This function defines the parameters for a Segment ARB® waveform.

Format

```
int seg_arb_define(INSTR_ID inst_id, long ch, long nsegments,
double *startvals, double *stopvals, double *timevals, long
*triggervals, long *outputRelayVals);
```

`instr_id`	`Instrument ID of the PG2: VPU1, VPU2, and so on.`
`ch`	The PG2 channel: 1 or 2.
`nsegments`	The number of values in each of the arrays (1024 maximum).
`startvals`	An array of start voltage values for each segment (in volts).
`stopvals`	An array of stop voltage values for each segment (in volts).
`timevals`	An array of time values for each segment (20 ns minimum).
`triggervals`	An array of trigger values: 0 (trigger low) or 1 (trigger high).
`outputRelayVals`	An array of values to control the high endurance output relay: 0 (open) or 1 (closed).

Remarks Each channel of the Model 4205-PG2 can be configured to output its own unique Segment-Arb waveform. A Segment ARB waveform is made up of user-defined segments (up to 1024). Each segment can have a unique time interval, start value, stop value, output trigger level (TTL high or low) and output relay state (open or closed).

> ***NOTE*** *The Model 4200-PG2 does not have high-endurance output relays (HEOR). When this function is used for an upgraded Model 4200-PG2, the values for the relays will be ignored.*

See Segment ARB in Section 11 for details on this pulse mode. The following arrays are required for the example Segment ARB® waveform shown in Figure 11-3:

Table 8-20
seg_arb_define

Start	Stop	Time	Trigger	Output relay
startvals(0) = 0.0	stopvals(0) = 1.0	timevals(0) = 50 e-9	triggervals(0) = 1	outputRelayVals(0) = 0
startvals(1) = 1.0	stopvals(1) = 1.0	timevals(1) = 100 e-9	triggervals(1) = 1	outputRelayVals(1) = 0
startvals(2) = 1.0	stopvals(2) = 1.5	timevals(2) = 20 e-9	triggervals(2) = 1	outputRelayVals(2) = 0
startvals(3) = 1.5	stopvals(3) = 1.5	timevals(3) = 150 e-9	triggervals(3) = 0	outputRelayVals(3) = 0
startvals(4) = 1.5	stopvals(4) = 0.0	timevals(4) = 50 e-9	triggervals(4) = 0	outputRelayVals(4) = 0
startvals(5) = 0.0	stopvals(5) = 0.0	timevals(5) = 500 e-9	triggervals(5) = 0	outputRelayVals(5) = 0
startvals(6) = 0.0	stopvals(6) = 0.0	timevals(6) = 130 e-9	triggervals(6) = 0	outputRelayVals(6) = 1

See also **`arb_file`**, **`arb_array`**, **`seg_arb_file`**

seg_arb_file Loads a waveform from a Segment ARB waveform file

Purpose

This function is used to load a waveform from an existing Segment ARB® waveform file.

Format

```
int seg_arb_file(INSTR_ID instr_id, long ch, char *fname)
```

`instr_id`	Instrument ID of the PG2: VPU1, VPU2, and so on.
`ch`	The PG2 channel: 1 or 2.
`fname`	The name of the waveform file.

Remarks

This function is used to load a waveform from an existing Segment ARB `.ksf` waveform file into the pulse generator card. A Segment ARB waveform can be loaded for each channel of the pulse generator card. Once loaded, use `pulse_output` to turn on the appropriate channel, and then use `pulse_trig` to select the trigger mode and start (or arm) pulse output.

When specifying the `fname`, include the full command path with the file name. Existing `.ksf` waveforms are typically saved in the `SarbFiles` folder at the following command path location:

`C:\S4200\kiuser\KPulse\SarbFiles`

A Segment ARB waveform can be created using KPulse, and then saved as a `.ksf` waveform file (refer to Section 13 for details).

A waveform in an existing `.ksf` file can be modified using a text editor.

NOTE *The Model 4200-PG2 does not have high-endurance output relays (HEOR). When this function is used for an upgraded Model 4200-PG2, the file will load, but the values for the relays will be ignored when the output is turned on.*

See also **arb_array**, **arb_file**, **seg_arb_define**

Example

The following function loads a Segment ARB file named `sarb3.ksf` (saved in the `SarbFiles` folder) into the pulse generator card for channel 1:

```
seg_arb_file(VPU1, 1,
C:\\S4200\\kiuser\\KPulse\\SarbFiles\\sarb3.ksf)
```

LPT functions for the Models 4220-PGU and 4225-PMU

NOTE *The Model 4200-SCS has built-in project plan tests that use functions documented below. The PMU-DUT-Examples project in Section 16 provides simple examples for coding a PMU user test module (UTM).*

The following information explains the functions included in the Keithley Linear Parametric Test Library (LPTLib) for the Models 4220-PGU and 4225-PMU. To simplify the descriptions of these two pulse cards in this documentation, the model names are abbreviated as PGU (pulse-generator unit) and PMU (pulse-measure unit).

The difference between these two pulse cards is that the PGU functions only as a pulse generator, but the PMU has both pulse and measurement capabilities. See PGU (pulse only) and PMU (pulse and measure)* for a summary of the functions for the PGU and PMU.

NOTE *The PGU and PMU support all the pulsing functions that the Model 4205-PG2 can perform. See PG2 (pulse only)* for a summary of the PG2 functions.*

The PGU and PMU support the following pulse modes:

- Standard pulse mode: For this two-level pulse mode, the user defines an amplitude and base level for the pulse output.
- Segment ARB pulse mode: For this multi-level pulse mode, you define a pulse waveform that consists of three or more line segments. Segment ARB pulse mode for the PGU and PMU also includes sequencing and sequence looping (see seg_arb_sequence and seg_arb_waveform for the PGU and PMU).
- Full arb pulse mode: For this multilevel pulse mode, the waveform consists of a number of user-defined points (see arb_array and arb_file).

Use the following instrument ID (identification) for LPT functions for the PGU and PMU:

- Model 4220-PGU: The instrument ID is VPU (VPU1, VPU2, and so on)
- Model 4225-PMU: The instrument ID is PMU (PMU1, PMU2, and so on)

The LPT functions for the PGU and PMU, which are described in detail on the following pages, include:

dev_abort (PGU, PMU)
PostDataDouble (PMU)*
PostDataDoubleBuffer (PMU)
pulse_chan_status (PMU)
pulse_conncomp (PMU)
pulse_exec (PGU, PMU)
pulse_exec_status (PGU, PMU)
pulse_fetch (PMU)
pulse_chan_status (PMU)
pulse_limits (PMU)
pulse_meas_sm (PMU)
pulse_meas_wfm (PMU)
pulse_meas_timing (PMU)
pulse_measrt (PMU)
pulse_ranges (PGU, PMU)
pulse_remove (PGU, PMU)
pulse_sample_rate (PMU)
pulse_source_timing (PGU, PMU)
pulse_step_linear (PGU, PMU)
pulse_sweep_linear (PGU, PMU)
pulse_train (PGU, PMU)
rpm_config (PMU)
seg_arb_sequence (PGU, PMU)
seg_arb_waveform (PGU, PMU)
setmode (PMU)

* The PostDataDouble function is also used with the SMU and CVU (see Enabling real time plotting for UTMs).

dev_abort Aborts a pulse_exec test

Purpose This function is used to programmatically end a test from within the user module (abort a test) that was started with the `pulse_exec` command.

Pulsers 4220-PGU (VPU), 4225-PMU

Pulse mode Standard and Segment ARB

Format
```
int dev_abort (NULL);
```

Remarks Use this function to abort a PGU or PMU pulse test from within a user module. This command is useful during a longer `pulse_exec` test. Note that `pulse_exec` is nonblocking, so it is possible to fetch data during a longer test. An example of this is a long stress/measure test, using `seg_arb_sequence` and `seg_arb_waveform`, that evaluates degradation data intra-test. Evaluating this data from within the user module

may determine that a test should end. For example, if the degradation is greater than ten percent (> 10%), then end the test (saves test time). This command is necessary to properly abort any test that uses `pulse_exec`, from within the user module.

Example The following code example illustrate placement of this command in a code a fragment. Note that after the `dev_abort` command is sent it is still necessary to use `pulse_exec_status` to poll and wait for the test to be ended.

```
// Place code to configure the PMU test here
//
// Start the test (for seg-arb testing, or for standard pulsing
// with no ranging, LLEC, or I/V/P threshold detection)
status = pulse_exec(PULSE_MODE_SIMPLE);
         if ( status )
         {
              // Minimal error handling, release memory used to
              // fetch results and stop test
              Free_Used_Arrays();
              return status;
         }
// loop to fetch data, while waiting for test complete
abort_sent = 0;
while(pulse_exec_status(&elapsedt) == 1)
      {
           // Code to fetch and evaluate data here
           if (abort_sent == 0)
           {
                // Code to fetch PMU data
                // Code to evaluate data
                // Code to determine if an abort is required
           }

           // If the test must be aborted, send dev_abort
           if (abort_required && abort_sent == 0)
           {
                dev_abort(NULL);
                abort_sent = 1;
           }
           Sleep(100);
      }
```

PostDataDouble Post buffer data into KITE Sheet tab
PostDataDoubleBuffer Post buffer data into KITE Sheet tab (for large data sets)

Purpose

`PostDataDouble`	Post data retrieved from the buffer into the KITE Sheet tab.
`PostDataDoubleBuffer`	Post PMU data retrieved from the buffer into the KITE Sheet tab (large data sets).

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Formats

PostDataDouble

```
int PostDataDouble(char *ColName, double *array);
```

`*ColName`	Column name for the data array into the KITE Sheet tab.
`array`	An array of data values for the KITE Sheet tab

PostDataDoubleBuffer

```
int PostDataDoubleBuffer(char *ColName, double *array, int
length);
```

`*ColName`	Column name for the data array in the KITE Sheet tab.
`array`	An array of data values for the KITE Sheet tab.
`length`	Number of data points (up to 65,535) to post into the KITE Sheet tab.

Remarks

The `PostDataDouble` and `PostDataDoubleBuffer` functions are used to post double precision, floating point PMU data into the KITE Sheet tab. Up to 65,535 points (rows) of data can be posted into the Sheet tab.

NOTE *The PostDataDouble function can also be used with the SMU and CVU (see Enabling real time plotting for UTMs).*

NOTE *The PostDataDoubleBuffer function is not compatible using KXCI to call user libraries remotely (see Calling KULT user libraries remotely in Section 9. Use PostDataDouble for user routines (UTMs) that will be called via KXCI.*

Either of these two functions can be used to post PMU data into the Sheet tab. However, the `PostDataDoubleBuffer` function should be used to post large data sets typically generated by PMU waveform measurements.

The `PostDataDouble` function uses two parameters. The `array` parameter is the array of data values to be posted into the Sheet tab. The **ColName* parameter is used to name the column for the data array into the Sheet tab.

The `PostDataDoubleBuffer` function is similar but uses a third parameter. The `length` parameter specifies the number of data points to post into the Sheet tab.

The following sequence summarizes the process to post data into the Sheet tab:

- Run a test.
- Use the pulse_fetch function to retrieve the data from the buffer. If desired, the retrieved data can be analyzed and/or manipulated.
- Use the `PostDataDouble` or `PostDataDoubleBuffer` function to post data into the KITE Sheet tab.

When using the `pulse_fetch` function, you can wait until the test is finished before retrieving data or you can retrieve blocks of data while the test is running, which is useful for a test that takes a long time. Instead of waiting for the entire test to finish, you can retrieve blocks of data at prescribed intervals. For details, see Data retrieval options for pulse_fetch.

NOTE *If you do not need to analyze and/or manipulate the test data before posting it into the Sheet tab in KITE, you can use the pulse_measrt function. The*

`pulse_measrt` *function retrieves all the test data in pseudo real-time and automatically posts it into the KITE Sheet tab.*

Examples

Two examples are provided. One uses the `PostDataDouble` function to post spot mean measurement data into the KITE Sheet tab. The second example uses the `PostDataDoubleBuffer` function to post waveform measurement data into the Sheet tab.

PostDataDouble example

This example assumes that a PMU spot mean test is configured to perform 100 (or more) voltage and current measurements for pulse high and low. Use the pulse_meas_sm function to configure the spot mean test.

The code below performs the following tasks:

- Starts the configured test.
- Uses a while loop to allow the spot mean test to finish.
- Retrieves voltage and current readings (100 data points) from the buffer.
- Separates the voltage and current readings for high (amplitude) and low (base).
- Posts the high measurement data into the KITE Sheet tab. Low measurement data is not posted into the Sheet tab.

```
// Code to configure the PMU test here

// Start the test (no analysis)
pulse_exec(0);
// While loop (continues while test is still running), with delay
// (30 ms)
while(pulse_exec_status(&elapsedt) == 1)
{
   Sleep(30);
}
// Retrieve V and I data (no timestamp or status)
status = pulse_fetch(PMU1, 1, 0, 100, Vmeas, Imeas, NULL, NULL);
// Separate V & I measurements for high (amplitude) and
// low (base)
for (i = 0; i<100; i++)
{
   VmeasHi_sheet[i] = Vmeas[2*i];
   ImeasHi_sheet[i] = Imeas[2*i];
   VmeasLo_sheet[i] = Vmeas[2*i+1];
   ImeasLo_sheet[i] = Imeas[2*i+1];
   PostDataDouble("DrainVmeas", VmeasHi_sheet[i]);
   PostDataDouble("DrainImeas", ImeasHi_sheet[i]);
}
```

PostDataDoubleBuffer example

This example assumes that a PMU waveform test is configured to perform 20,000 (or more) voltage and current measurements. Use the pulse_meas_wfm function to configure the waveform test.

The code below performs the following tasks:

- Starts the configured test.

- Uses a while loop to allow the waveform test to finish.
- Retrieves voltage, current, and timestamp readings (20,000 data points) from the buffer.
- Separates the voltage, current, and timestamp readings.
- Posts the measurement data into the KITE Sheet tab.

```
// Code to configure the PMU test here
// Start the test (no analysis)
status = pulse_exec(0);
// While loop (continues while test is still running), with
// delay (30 ms)
while (pulse_exec_status(&elapsedt) == 1)
{
   Sleep(30);
}
// Retrieve V, I, and timestamp data (no status)
status = pulse_fetch(PMU1, 1, 0, 20e3, Vmeas, Imeas, Tstamp, NULL);
// Separate V, I, and timestamp measurements
for (i = 0; i<20e3; i++)
{
   Vmeas_sheet[i] = Vmeas[2*i];
   Imeas_sheet[i] = Imeas[2*i];
   Tstamp_sheet[i] = Tstamp[2*i];
}
PostDataDoubleBuffer("DrainVmeas", Vmeas_sheet, 20e3);
PostDataDoubleBuffer("DrainImeas", Imeas_sheet, 20e3);
PostDataDoubleBuffer("Timestamp", Tstamp_sheet, 20e3);
```

pulse_chan_status Returns the number of readings in the data buffer

Purpose This function is used to determine how many readings are stored in the data buffer.

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_chan_status(INSTR_ID instr_id, int chan, int
*buffersize);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	PMU channel: 1 or 2.
`*buffersize`	User-defined name for the returned value.

Remarks Use this function to return the number of readings presently stored in the data buffer. This function can be called while a sweep is in progress or after it is completed.

For a short sweep (test time in seconds to a minute or more), this function is typically called after the sweep is completed to determine the total number of readings stored in the buffer. For a long test, you can use this function to track the progress of the test. A long test is typically Segment ARB with test time in minutes, hours, or days).

Example This function returns the number of readings stored in the buffer for channel 1:

```
pulse_chan_status(PMU1, 1, buffersize);
```

pulse_conncomp Control connection compensation

Purpose This function controls (enables or disables) connection compensation.

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_conncomp(INSTR_ID instr_id, int chan, int type, int index);
```

`instr_id` Instrument ID: PMU1, PMU2, and so on.

`chan` PMU channel: 1 or 2.

`type` Type of compensation to enable:
1: Short
2: Delay

`index` 0: Disable (for both types)
Short:
1: Chx-Short-Res-2m-PMU
2: Chx-Short-Res-2m-RPM
3: Chx-Short-Res-2m-Custom
Delay:
1: Chx-Delay-2m-PMU
2: Chx-Delay-2m-RPM
3: Chx-Delay-2m-Custom
Where: x is the channel number (1 or 2)

Remarks Errors caused by the connections and cable length between the Model 4225-PMU and the device under test (DUT) can be corrected by using connection compensation. When connection compensation is enabled, each DUT measurement factors in either the default or measured (custom) compensation values.

Use this function to control connection compensation. There are two types of compensation using this function: Short and delay. Short compensation corrects for the measured resistance of the cabling and connections; delay compensation measures and corrects for cable delay (which is the time it takes a signal to transit the cable).

You have the option to use either default connection compensation values (PMU or RPM) or custom connection compensation values. The default values provide compensation for simple connection setups that use the supplied cables. The custom connection compensated values are generated when connection compensation is performed. The custom values provide optimum compensation.

`index` parameter values:

- 0: Disables all connection compensation.
- 1: Selects the default connection compensation values for a setup that uses the PMU only.
- 2: Selects the default connection compensation values for a setup that uses the PMU and the RPM.
- 3: Selects the custom connection compensation values.

Custom connection compensation is a two-part process:

1. Perform connection compensation from the KITE interface (see Performing connection compensation in Section 16). Connection compensation data is

generated for short and delay conditions. The compensation values are stored in tables.

2. Use this function (`pulse_conncomp`) to select (enable) the custom connection compensation values.

When a test is run, each measurement will factor in the enabled compensation values. If connection compensation is disabled, the compensation values will not be used by the test.

Example Assuming connection compensation was performed from the KITE interface, this function selects (enables) short connection compensation using the custom correction values:

```
pulse_conncomp(PMU1, 1, 1, 3);
```

pulse_exec Starts test execution

Purpose This function is used to validate the test configuration and start test execution.

Pulsers Models 4220-PGU (VPU) and 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_exec(long mode);
```

`mode`

`PULSE_MODE_SIMPLE` or 0: No analysis performed during testing; no ranging, no load line effect compensation, and no threshold checking.

`PULSE_MODE_ADVANCED` or 1: Enables the analytical sweep engine and incorporates the use of any combination of the various options for the standard (2-level) pulse mode.

Remarks Use this function to validate the test configuration, select the simple or advanced mode, and execute the test. If there are any problems with the test configuration, the validation will stop and the test will not be executed.

The `pulse_exec` function is nonblocking, which means that if this function is called to execute the test, the program will continue and not wait for the test to finish. Therefore, after calling pulse_exec, the pulse_exec_status command must be called in a while loop to ensure the test is complete before fetching data or exiting the UTM.

When using pulse_fetch to retrieve data, you need to pause the program to allow time for the buffer to fill. You can use the sleep function to pause for a specified period of time, or you can use the pulse_exec_status function in a `while` loop to wait until the test is completed.

There are two functions that will affect a pulse test while it is running:

- The pulse_remove function removes a PMU channel from the test.
- The dev_abort function aborts the test.

NOTE *The "Internal Trigger Bus" trigger source (see pulse_trig_source function) is used only by the Models 4220-PGU and 4225-PMU for triggering. The pulse_exec command automatically uses the internal trigger bus.*

CAUTION	**Do not exit the user module while the test is still running, or incorrect readings or device damage may result.**

Example Program fragment 1 shows how to use `pulse_exec` which validates the test configuration, sets the execution type to simple two-level pulse operation (no analysis), and executes the test.

pulse_exec_status Checks the run status of the test

Purpose This function is used to determine if a test is running or completed.

Pulsers Models 4220-PGU (VPU) and 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_exec_status(double *elapsedt);
```

`*elapsedt` Name of the user-defined pointer for elapsed time.

Remarks This function is required to determine when a test is complete or what is occurring during a test. The return value indicates whether the test is still running (`PMU_TEST_STATUS_RUNNING` or `1`) or completed (`PMU_TEST_STATUS_IDLE` or `0`). The primary use of this command is to ensure that the test is completed before fetching PMU data or ending the test. See the code example link below. The parameter for this function is a pointer to indicate elapsed time.

The elapsed time is the KITE test time, not the PMU or VPU card test time. For short test times, the returned elapsed time will be longer than the actual time required on-card.

This function is typically used in a `while` loop to allow the test to finish before retrieving the data using the pulse_fetch function (see Program fragment 1).

It is the responsibility of the UTM programmer to ensure that the pulse test is complete before exiting the UTM. If the UTM program ends before the test is complete, KITE will respond with two messages. These messages are displayed in the KITE messages window. Five seconds after the UTM ends prematurely (before the pulse test is finished), the message "UTMname ended before the test was complete. Waiting for test to finish (max wait = 5 minutes).". KITE will continue to wait for the UTM to finish, interrupting further test execution. After the default of five minutes, the UTM is terminated and the following message is displayed, "UTMname did not finish before the maximum wait period. UTM aborted." After this five minute wait, KITE will release control to the GUI, or the next test in the project (if using repeat executing or looping).

See also **`pulse_exec`**

Example Program fragment 1 shows how to use `pulse_exec_status` in a `while` loop.

pulse_fetch Retrieves enabled test data

Purpose This function retrieves enabled test data and temporarily stores it in the data buffer.

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_fetch(INSTR_ID instr_id, int chan, int StartIndex, int
StopIndex, double *Vmeas, double *Imeas, double *Timestamp,
unsigned long *Status);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	PMU channel: 1 or 2.
`StartIndex`	Start index point for data (within the overall set of data).
`StopIndex`	Final index point to be retrieved.
`*Vmeas`	Name of the user-defined array for retrieved voltage measure readings. This is a single-dimension array.
`*Imeas`	Name of the user-defined array for retrieved current measure readings. This is a single-dimension array.
`*Timestamp`	Name of the user-defined array for retrieved time stamps. This is a single-dimension array.
`*Status`	Name of the user-defined array for retrieved status for the channel.

Remarks Use this function to retrieve a block of newly generated test data in pseudo real-time and temporarily store it in the data buffer. The stored data can then be analyzed and manipulated as needed before posting it to the Sheet tab in KITE.

Typically, this function is used with the `pulse_exec_status` function to allow the test to finish before retrieving the data. For details, see Wait until the test is completed before retrieving data.

The block of data to be retrieved is set by the `StartIndex` and `StopIndex` parameters. The start index parameter specifies the first index number in the buffer, and the stop index parameter specifies the final index number. For example, assume there are 1000 data test points for a test, and you want to retrieve the first 50 data points. The start index value is set to zero (0) and the stop index is set to 49.

The `*Vmeas`, `*Imeas`, `*Timestamp`, and `*Status` parameters are array names defined by the user. If you do not want to retrieve the time stamp or status, `NULL` can be passed as a valid parameter for this field.

NOTE *The return of all readings must be enabled by the pulse_meas_sm function. If disabled, the arrays will not be retrieved.*

For spot mean measurements, amplitude and base level readings are returned in the same array buffer area and must be separated (or parsed) after the measurement cycle is complete. See pulse_meas_sm function for details on spot mean measurements. Note that number of measurements returned is determined by the spot means enabled in pulse_meas_sm. With both amplitude and base measurements enabled, there will be two voltage and two current readings for each pulse (with spot mean discrete) or each pulse burst (with spot mean average). Voltage and current readings are returned in individual arrays: Vmeas, Imeas. When both amplitude and base readings are enabled, the readings are alternated. For example, the Vmeas array: Vampl_1, Vbase_1, Vampl_2, Vbase_2, Vampl_3, Vbase_3, etc. To plot the amplitude values, separate the amplitude and base measurements into individual arrays before using PostDataDouble to post the measurements to the sheet.

The time stamps pertain to either per spot mean reading or per sample. Status is returned as a 32-bit word. The status code bit map is shown in Table 8-21.

Table 8-21
Status-code bit map for pulse_fetch

Bit	Summary or description	Value (bit pattern)
31	Reserved	Reserved bit for future use
30	Sweep skipped	0 = Not skipped 1 = Skipped
29	Load line effect compensation (LLEC) enabled (only valid when LLEC is enabled)	0 = Failed 1 = Successful
28	LLEC status	0 = Disabled 1 = Enabled
27 - 24	RPM mode settings	0 (0000) = No RPM 1 (0001) = RPM 2 (0010) = Bypass; PMU 3 (0011) = Bypass; SMU 4 (0100) = Bypass; CVU All other values (bit patterns) reserved
23 - 20	Reserved	Reserved bits for future use
19 - 16	Measurement type	1 (0001) = Spot mean 2 (0010) = Waveform All other values (bit patterns) reserved
15 -12	Current threshold, voltage threshold, power threshold, and source compliance	0 (0000) = None 1 (0001) = Source compliance 2 (0010) = Current threshold reached or surpassed 4 (0100) = Voltage threshold reached or surpassed 8 (1000) = Power threshold reached or surpassed
11 - 10	Current measure overflow	0 (00) = No overflow 1 (01) = Negative overflow 2 (10) = Positive overflow
9 - 8	Voltage measure overflow	0 (00) = No overflow 1 (01) = Negative overflow 2 (10) = Positive overflow
7 - 4	Current measure range	0 (0000) = 100 nA (RPM only) 1 (0001) = 1 µA (RPM only) 2 (0010) = 10 µA (RPM only) 3 (0011) = 100 µA 4 (0100) = 1 mA (RPM only) 5 (0101) = 10 mA 6 (0110) = 200 mA 7 (0111) = 800 mA All other values (bit patterns) reserved
3 - 2	Voltage measure range	0 (00) = 10 V 1 (01) = 40 V
1 - 0	Channel number	1 (01) = Ch1 2 (10) = Ch2 Value 0 (00) not used

NOTE *If you do not need to analyze and/or manipulate the test data before posting it to the Sheet tab in KITE, you can use the pulse_measrt function. The* `pulse_measrt` *function retrieves all the test data in pseudo real-time and automatically posts it into the KITE Sheet tab.*

Data retrieval options for pulse_fetch

There are two options to retrieve data: 1) Wait until the test is completed, or 2) retrieve blocks of data while the test is running.

Because pulse_exec is a non-blocking function, the running user test module (UTM) will continue after it is called to start the test. This means that the program will not automatically pause to allow the pulse-measure test to finish.

CAUTION **The programmer must ensure that the test program does not finish or return to KITE before the test is complete, or erroneous results and damage to test devices may occur.**

If `pulse_fetch` is inadvertently called before the test is completed, the data buffer may not fill with all the requested readings. Array entries are designated as zero for test data that is not yet available.

Wait until the test is completed before retrieving data

An effective method to pause the program is to monitor the status of the test by using a `while` loop to check the returned value of pulse_exec_status. When the test is completed, the program drops out of the loop and calls `pulse_fetch` to retrieve all the test data. The following program fragment shows how to use a `while` loop.

Program fragment 1

```
// Code to configure the PMU test here

// Start the test (no analysis)
pulse_exec(0);

// while loop and short delay (10 ms)

while (pulse_exec_status(&elapsedt) == 1)

{

   Sleep(10);

}

// Retrieve all data
status = pulse_fetch(PMU1, 1, 0, 49, Drain_Vmeas, Drain_Imeas,
NULL, NULL);

// Code for data handling here
```

After all the data is retrieved, it can be analyzed, manipulated and then posted into the KITE Sheet tab. Use the PostDataDouble or PostDataDoubleBuffer function to post the data.

Retrieve blocks of data while test is running

An advantage of the `pulse_exec` function being non-blocking is that it allows you to retrieve test data before the test is completed, which is useful for a test that takes a long time. Instead of waiting for the entire test to finish, you can retrieve blocks of data at prescribed intervals. The interval can be controlled by using the `sleep` function as shown in the following program fragment.

Program fragment 2

```
// Code to initialize the data arrays
for i (i = 0; i < array_size; i++)
{
   Drain_Vmeas = 0.0;
   Drain_Imeas = 0.0;
}


// Code to configure the PMU test here
// Start the test and pause for 20 seconds
pulse_exec(0);
Sleep(20000);
// Retrieve a block of test data:
pulse_fetch(PMU1, 1, 0, 10e3, Drain_Vmeas, Drain_Imeas, 1,
NULL);
// Code for data handling here
```

After retrieving a block of data, loop back to the sleep function to allow the next block of data to become available before fetching it. Repeat this loop until all the data is retrieved.

The `pulse_fetch` command will return all data available at the time of the call. The remaining array space will not be modified. To determine how much data was retrieved, it is recommended to initialize the arrays. **Program fragment 2** initializes the results arrays to 0.0, but other values may be used. After the retrieving the data, search the array for the first entry with this initialized value.

Retrieved blocks of data can be analyzed and manipulated while the test is still running. After data handling is completed, use the PostDataDoubleBuffer function to post the data to the KITE Sheet tab.

Example

This function retrieves 50 points of data from the buffer:

```
pulse_fetch(PMU1, 1, 0, 49, Drain_Vmeas, Drain_Imeas, T_Stamp,
NULL);
```

Where:

`Instr_id` = `PMU1`
`chan` = `1` (channel 1)
`StartIndex` = `0`
`StopIndex` = `49`
`Vmeas` = `Drain_Vmeas` (name of array)
`Imeas` = `Drain_Imeas` (name of array)
`Timestamp` = `T_Stamp` (name of array)
`Status` = `NULL` (not retrieved)

pulse_limits Sets voltage, current, and power thresholds

Purpose

Sets measured voltage and current thresholds at the DUT. Also sets the power threshold for each channel.

Pulsers Model 4225-PMU

Pulse mode Standard

Format

```
int pulse_limits(INSTR_ID instr_id, int chan, double V_Limit,
double I_Limit, double Power_Limit);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	Pulse generator channel: 1 or 2.
`V_Limit`	Measured voltage (V) threshold at the DUT.
`I_Limit`	Measured current (A) threshold at the DUT.
`Power_Limit`	Power (W) threshold for the channel (Power = Vmeas x Imeas).

Remarks This feature is different from a SMU compliance setting, in that threshold checking is performed after each burst of pulses, using the spot mean values to compare to the specified thresholds. The thresholds are checked against all enabled measurements for the channel. If a threshold is reached or exceeded, the present sweep is stopped and testing continues with any subsequent sweeps.

This feature doesn't prevent the set thresholds from being reached or exceeded. After detecting a threshold breach, it simply aborts the sweep.

Maximum power for each PMU source range:

High-speed voltage source (10 V) range: Maximum power = 5 V x 0.1 A = 0.5 W
High-voltage source (40 V) range: Maximum power = 20 V x 0.4 A = 8 W

Example This function sets thresholds for channel 1 of the PMU:

```
pulse_limits(PMU1, 1, 42, 1, 10);
```

`Instr_id` = PMU1
`chan` = 1 (channel 1)
`V_Limit` = 42 V
`I_Limit` = 1 A
`Power_Limit` = 10 W

pulse_meas_sm Configures spot mean measurements
pulse_meas_wfm Configures waveform measurements
pulse_meas_timing Sets the measurement windows

Purpose

`pulse_meas_sm`	This function configures spot mean measurements.
`pulse_meas_wfm`	This function configures waveform measurements.
`pulse_meas_timing`	This function sets the measurement windows.

Pulsers Model 4225-PMU

Pulse mode Standard

Formats **pulse_meas_sm**

```
int pulse_meas_sm(INSTR_ID instr_id, int chan, Int AcquireType,
int AcquireMeasVAmpl, int AcquireMeasVBase, int
AcquireMeasIAmpl, int AquireMeasIBase, int AquireTimeStamp, int
LLEComp);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	PMU channel: 1 or 2.
`AcquireType`	Acquisition type: 0 = Discrete 1 = Average
`AcquireMeasVAmpl`	Enable (1) or disable (0) return of amplitude voltage measurements.
`AcquireMeasVBase`	Enable (1) or disable (0) return of base level voltage measurements.
`AcquireMeasIAmpl`	Enable (1) or disable (0) return of amplitude current measurements.
`AcquireMeasIBase`	Enable (1) or disable (0) return of base current level measurements.
`AcquireTimeStamp`	Enable (1) or disable (0) return of time stamp readings.
`LLEComp`	Load line effect compensation (LLEC): 0 = All LLEC disabled 1 = Voltage LLEC on for pulse amplitude only LLEC is only performed for standard pulse IV testing using PMU measure ranges. It is not performed when using Model 4225-RPM measure ranges. The active RPM circuitry provides its own analog LLEC (assuming a short cable from the RPM to the DUT).

pulse_meas_wfm

```
int pulse_meas_wfm(INSTR_ID instr_id, int chan, int AcquireType,
int AcquireMeasV, int AcquireMeasI, int AquireTimeStamp, int
LLEComp);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	PMU channel: 1 or 2.
`AcquireType`	Acquisition type: 0 = Discrete 1 = Average
`AcquireMeasV`	Enable (1) or disable (0) return of voltage measurements.
`AcquireMeasI`	Enable (1) or disable (0) return of current measurements.
`AcquireTimeStamp`	Enable (1) or disable (0) return of time stamp readings. Time stamp must be enabled in order to measure waveforms.
`LLEComp`	Load line effect compensation (LLEC): 0 = LLEC disabled 1 = LLEC enabled LLEC is only performed for standard pulse IV testing using PMU measure ranges. It is not performed when using Model 4225-RPM measure ranges. The active RPM circuitry provides its own analog LLEC (assuming a short cable from the RPM to the DUT).

pulse_meas_timing

```
int pulse_meas_timing(INSTR_ID instr_id, int Chan, double,
StartPercent, double StopPercent, int NumPulses);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	PMU channel: 1 or 2
`StartPercent`	Spot mean: Start location for spot mean measurements, specified as a percentage of the widths for the amplitude and base level (see Figure 8-117). Waveform: Pre-data for the amplitude, specified as a percentage of the amplitude pulse duration (see Figure 8-121).
`StopPercent`	Spot mean: Stop location of the spot mean measurements, specified as a percentage of the widths for the amplitude and base level (see Figure 8-117). Waveform: Post-data for the amplitude, specified as a percentage of the amplitude pulse duration (see Figure 8-121).
`NumPulses`	Number of pulses to output and measure (1 to 10,000).

NOTE *Use the pulse_sample_rate function to set the sampling rate for pulse measurements.*

Remarks

There are four types of pulse measurements that are performed by the Model 4225-PMU: Spot mean discrete, spot mean average, waveform discrete, and waveform average. Use the following pulse generator functions to configure pulse measurements:

- Use the `pulse_meas_sm` function to configure spot mean measurements; select the data acquisition type, set the readings to be returned, enable or disable time stamp, and set LLEC. See Spot mean measurements for details.
- Use the `pulse_meas_wfm` function to configure waveform measurements; select the data acquisition type, set the readings to be returned, enable or disable time stamp, and set LLEC. See Waveform measurements for details.
- Use the `pulse_meas_timing` function to set measurement timing. For spot mean measurements, portions of the amplitude and base levels are specified for sampling. For pre-data and post-data waveform measurements, a percentage of the entire pulse duration is specified. See Measurement timing for details on pulse measurement timing.

NOTE *Before calling the* `pulse_meas_timing` *function, use the* `pulse_meas_sm` *or* `pulse_meas_wfm` *function to configure the measurement type.*

Measurement types

There are two types of pulse measurements: Spot mean and waveform. The pulse_meas_sm function is used to configure Spot mean measurements and the pulse_meas_wfm function is used to configure Waveform measurements.

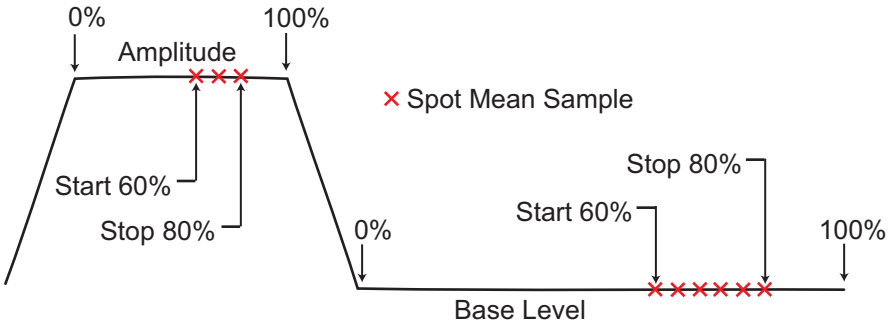
Spot mean measurements

Spot mean measurements sample a portion of the amplitude and a portion of the base level. The measured samples are then averaged to yield a single voltage and current reading for the amplitude and base low levels. The `NumPulses` (number of pulses) parameter is used to specify the number of pulses to be output and sampled. To return individual spot mean for each pulse, see Spot mean discrete readings. To have a single

spot mean for all NumPulses, see Spot mean average readings. With both amplitude and base spot means enabled, each pulse will have 2 voltage measurements and two current measurements. See pulse_fetch for details.

In Figure 8-117, three measured samples are taken on the amplitude and six samples are taken on the base level. The start and stop percentage values shown in Figure 8-117 indicate the portions of the pulse that are sampled. See Spot mean measurement timing for details on measurement timing.

Figure 8-117
Spot mean measurements example

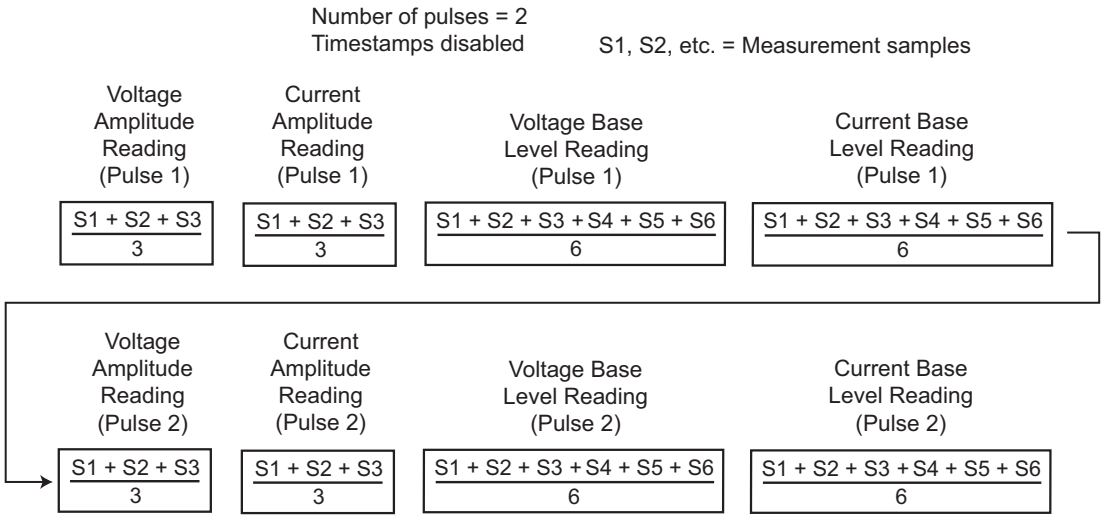


There are two data acquisition types of spot mean measurements; Spot mean discrete readings and Spot mean average readings.

Spot mean discrete readings

The averaged voltage and/or current readings for every sampled pulse period are returned in a single data set. Figure 8-118 shows how spot mean discrete readings are returned as a data set for two pulse periods. With all voltage and current readings enabled, four readings are returned for each pulse. The measured samples are averaged to yield the mean average readings. When time stamps are enabled, a time stamp is included in the data set after each mean reading.

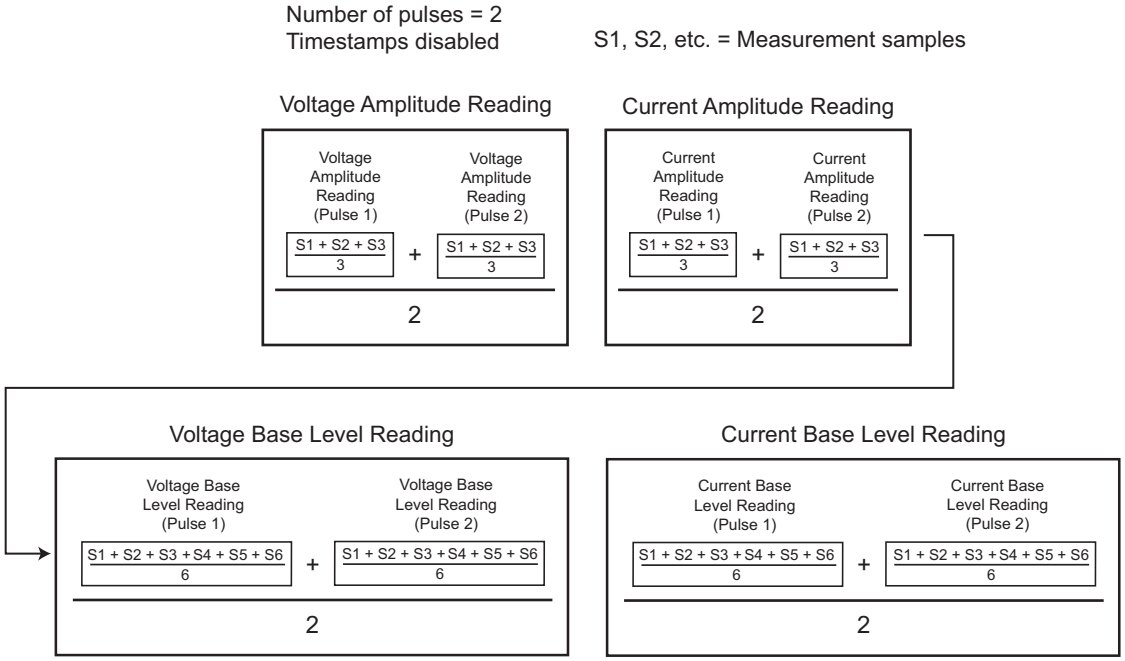
Figure 8-118
Returned data set for spot mean discrete readings



Spot mean average readings

For this data acquisition type, each returned reading is a mean-of-the-means. Spot mean average averages the mean readings for all the pulses in the burst. In Figure 8-118, each mean reading for Pulse 1 is averaged with each corresponding mean reading for Pulse 2 to yield the mean-of-the-means readings shown in Figure 8-119. With all voltage and current readings enabled, four readings are returned for the burst. When time stamps are enabled, a time stamp is included in the data set after each mean-of-the-means reading.

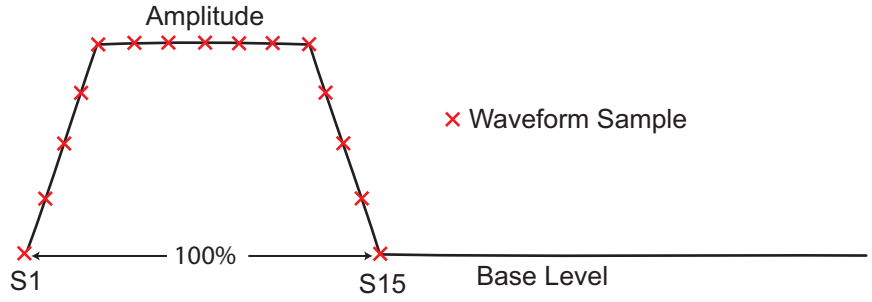
Figure 8-119
Returned data set for spot mean average readings



Waveform measurements

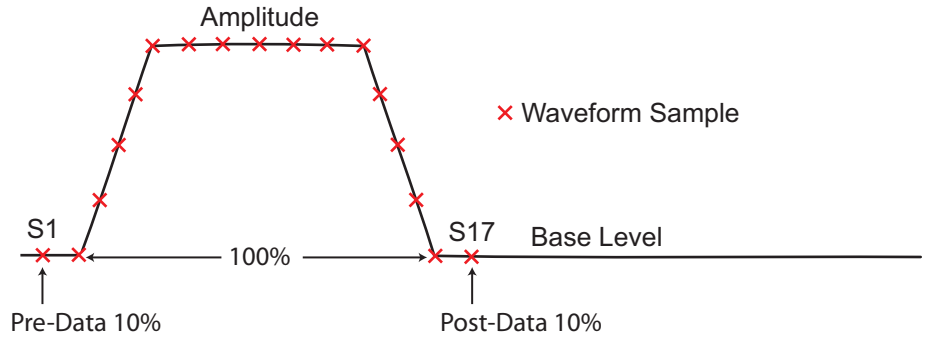
Waveform measurement readings sample the entire pulse. Sampling is performed on the rise time, top width, and fall time portions of the pulse. In Figure 8-120, 15 samples are performed on the pulse waveform. Enabled voltage and current readings and time stamps are returned for every sample taken on the pulse. The `NumPulses` (number of pulses) parameter is used to specify the number of pulses to be output and sampled.

Figure 8-120
Waveform measurements



A waveform measurement can include pre-data and post-data. Pre-data is extra data taken before the rise time of the pulse; post-data is extra data taken after the fall time (see Figure 8-121). See Waveform measurement timing for details on measurement timing for pre-data and post-data.

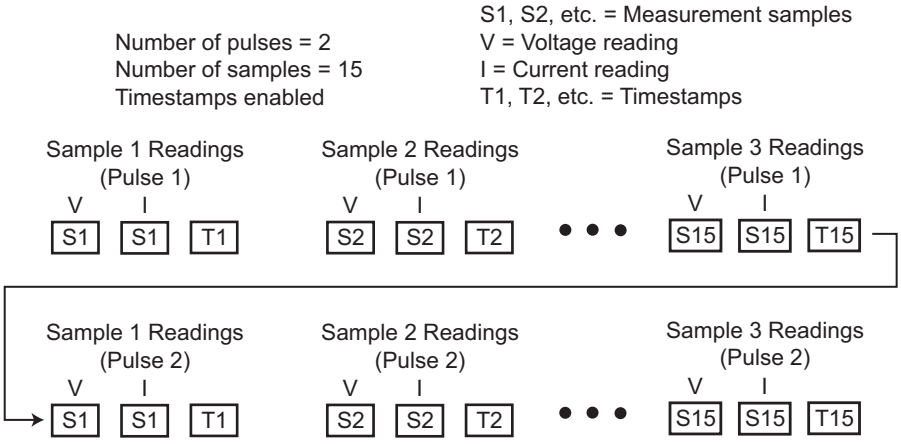
Figure 8-121
Waveform measurements (with pre-data and post-data)



Waveform discrete readings

Enabled voltage and/or current readings and time stamps for every sample of the waveform are returned in a single data set. Figure 8-122 shows how waveform discrete readings are returned as a data set for two pulse periods with voltage, current, and time stamp readings enabled.

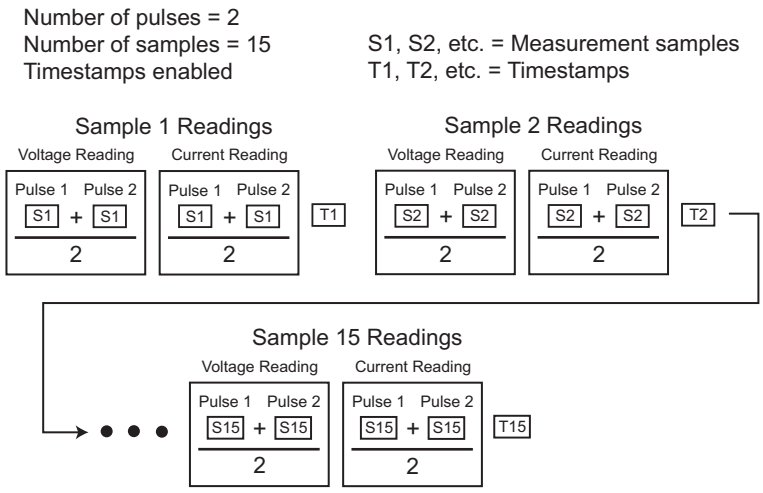
Figure 8-122
Returned data set for waveform discrete readings



Waveform average readings

For this data acquisition type, each returned reading is a mean average of the corresponding samples for all the pulses in the burst. For example, in Figure 8-122 the V and I readings for Sample 1, Pulse 1 would be averaged with the V and I readings for Sample 1, Pulse 2. Figure 8-123 shows how waveform average readings are returned as a data set for two pulse periods with voltage, current, and time stamp readings enabled.

Figure 8-123
Returned data set for waveform average readings



Load line effect compensation (LLEC)

The basic pulse test system is a series circuit that consists of the pulse card resistance (fixed at 50 Ω), interconnect resistance, and the resistance of the DUT. In this series circuit, the sum of the voltage drops across these three components is equal to the output voltage of the pulse generator. Therefore, if the resistance of the DUT changes, the voltage seen at the DUT also changes. This effect is called the “load line effect.” See DUT resistance determines pulse voltage across DUT in Section 11 for details on how the resistance of the DUT affects the voltage across it.

To counter load line effect, the Model 4225-PMU pulse generator has built-in load line effect compensation (LLEC). When LLEC is active, the pulse generator adjusts its output (within its limits) such that the programmed output voltage appears at the DUT.

LLEC uses a software algorithm to adjust the output of the pulser. When enabled, the algorithm performs a set number of iterations in an attempt to output the programmed output voltage to the DUT. The setmode function is used to set the number of iterations for the software algorithm.

NOTE *See Load line effect compensation (LLEC) for the PMU* in Section 16 for details on load line effect compensation.

The `LLEComp` parameter for the `pulse_meas_sm` and `pulse_meas_wfm` functions enables or disables LLEC. The following values can be set for the `LLEComp` parameter to control LLEC:

- Value 1 enables LLEC for amplitude only. It is disabled for base level.
- Value 0 disables LLEC for both amplitude and base level.

The magnitude of the pulse steps affects speed. The larger the magnitude, the longer it takes to perform LLEC.

There may be some high-impedance devices that will not test properly with LLEC enabled. In this case, you can disable LLEC (value 0).

NOTE *Another advantage of using LLEC is that it maintains linear voltage spacing during the test. For example, if the pulse sweep uses 200 mV steps, measurements will be performed at every 200 mV step. Data that is generated using linear voltage spacing is ready to be fed into a mathematical model. For details, see LLEC maintains even voltage spacing in Section 16.*

Measurement timing

The pulse_meas_timing function is used to configure pulse measurement timing

Spot mean measurement timing

The spot mean measurement type samples a portion of the amplitude and a portion of the base level. The portions to be sampled are specified as a percentage.

An example of spot mean measurements is shown in Figure 8-117. As shown, the beginning of the amplitude and base level are designated as the zero (0) percent points; the ends are designated as the 100 percent points. The start and stop points for amplitude and base-level sampling are expressed as a percentage between 0 and 100 percent. In Figure 8-117, sampling for the amplitude and base levels starts at 60 percent (0.6) and ends at 80 percent (0.8).

The number of samples taken on the amplitude and base level is dependent on the size of the portions to be sampled and the sampling rate. Use the pulse_sample_rate function to set the sampling rate for pulse measurements.

Waveform measurement timing

The waveform measurement type samples the pulse as shown in Figure 8-120. Sampling is performed on the entire duration (100 percent) of the pulse. This includes the rise time, amplitude, and fall time portions of the pulse. A voltage and/or current reading is returned for every sample.

A waveform measurement can include pre-data and post-data. Pre-data is extra data taken before the rise time of the pulse; post-data is extra data taken after the fall time. Figure 8-121 shows an example where 10 percent (0.1) pre-data and 10 percent (0.1) post-data is taken.

The number of samples taken on the pulse is dependent on the size of the pulse to be sampled and the sampling rate. Use the pulse_sample_rate function to set the sampling rate for pulse measurements.

NOTE *In the function, a percentage must be expressed as its decimal equivalent. For example, specify 50 percent as 0.5 in the function.*

Examples

pulse_meas_sm

This function sets channel 1 of the PMU for the spot mean discrete measure type to acquire the voltage amplitude measurement, the current base level measurement, and the time stamps. It also enables LLEC for amplitude:

```
pulse_meas_sm(PMU1, 1, 0, 1, 0, 1, 0, 1, 1);
```

Where:

`Instr_id` = PMU1
`chan` = 1 (channel 1)
`AcquireType` = 0 (discrete)
`AcquireMeasVAmpl` = 1 (enable)
`AcquireMeasVBase` = 0 (disable)

AcquireMeasIAmpl = 1 (enable)
AcquireMeasIBase = 0 (disable)
AcquireTimeStamp = 1 (enable)
LLEComp = 1 (enable)

pulse_meas_wfm

This function sets channel 1 of the PMU for the waveform discrete measure type to acquire the voltage/current readings for the waveform and the time stamps. It also enables LLEC.

```
pulse_meas_wfm(PMU1, 1, 0, 1, 1, 1, 1);
```

Where:

Instr_id = PMU1
chan = 1 (channel 1)
AcquireType = 0 (discrete)
AcquireMeasV = 1 (enable)
AcquireMeasI= 1 (enable)
AcquireTimeStamp = 1 (enable)
LLEComp = 1 (enable)

pulse_meas_timing

This function sets the following pulse measure timing settings for five spot mean measurements for channel 1 of PMU1:

```
pulse_meas_timing(PMU1, 1, 0.6, 0.8, 5);
```

Where:

Instr_id = PMU1
chan = 1 (channel 1)
StartPercent = 0.6 (60 percent)
StopPercent = 0.8 (80 percent)
NumPulses = 5 (output one pulse)

pulse_measrt Returns data in pseudo real-time

Purpose This function returns pulse source and measure data in pseudo real-time.

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_measrt(INSTR_ID instr_id, int chan, char
*VMeasColName, char *IMeasColName, char *TimeStampColName, char
*StatusColName);
```

instr_id	Instrument ID: PMU1, PMU2, and so on.
chan	Channel number: 1 or 2.
*VMeasColName	Column name for V-measure data in the KITE Sheet tab.
*ImeasColName	Column name for I-measure data in the KITE Sheet tab.
*TimeStampColName	Column name for the time stamp data in the KITE Sheet tab.
*StatusColName	Column name for the status data in the KITE Sheet tab.

Remarks Use this function to return and display test data in pseudo real-time. As measurements are performed, the data is automatically placed in the KITE Sheet tab. This function is

also used to name the columns in the KITE Sheet tab. This function must be called prior to calling pulse_exec to start the test.

NOTE *The pulse_measrt function is not compatible using KXCI to call user libraries remotely (see Calling KULT user libraries remotely in Section 9. Use PostDataDouble for user routines (UTMs) that will be called via KXCI.*

See also pulse_fetch

Example The following function configures channel 1 of PMU1 to return data in pseudo real-time:

```
pulse_measrt(PMU1, 1, "V-Measure", "I-Measure", "Timestamp",
"Status");
```

pulse_ranges Sets the pulse voltage and measure ranges

Purpose This function sets the voltage pulse range and voltage/current measure ranges.

Pulsers Models 4220-PGU and 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_ranges(INSTR_ID instr_id, int Chan, double VSrcRange,
int Vrange_type, double Vrange, int Irange_type, double IRange);
```

`instr_id` Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.

`Chan` Channel number: 1 or 2.

`VSrcRange` Source range: 5 or 20 (into 50 Ω)
10 or 40 (into ≥1 MΩ)

`Vrange_type` Voltage measure range type (PMU):
0 = Auto
1 = Limited auto
2 = Fixed

`Vrange_type` parameter is ignored by the PGU. Autorange (0) and limited autorange (1) are not valid for the Segment ARB pulse mode.

`Vrange` Voltage measure range (PMU)

`Vrange` parameter is ignored by the PGU and is ignored by the PMU when autorange is selected.

`Irange_type` Current measure range type (PMU):
0 = Auto
1 = Limited auto
2 = Fixed

`Irange_type` parameter is ignored by the PGU. Autorange (0) and limited autorange (1) are not valid for the Segment ARB pulse mode.

`Irange` Current measure range (PMU)

`Irange` parameter is ignored by the PGU and is ignored by the PMU when autorange is selected.

Remarks

For the PMU, use this function to (1) set the voltage source range for pulse output, (2) set the voltage and current measure range types, and (3) set the actual voltage and current measure ranges. For the PGU, use this function to set the voltage source range for pulse output.

Measure range types for the PMU:

- Fixed: Use this range type to specify a fixed measure range (`Vrange` or `Irange`).
- Limited Auto: Select this range type to use the fixed measure as the lowest range that will be used for automatic ranging.
- Auto: Use this range type to automatically select the optimum measure range. The specified fixed measure range (`Vrange` or `Irange`) is not used in auto-range mode, but must be a valid range.

Auto or limited auto-ranging is available only when using the advanced mode in the pulse_exec function. Ranging is controlled per-channel and may be combined with load line effect compensation (LLEC) and thresholds (see pulse_limits function for thresholds).

The Segment ARB pulse mode does not allow range changes (no autorange) within a Segment ARB® waveform definition. Only fixed ranging is available for the Segment ARB pulse mode.

Example

The following function sets the source-measure ranges for channel 1 of PMU1:

```
pulse_ranges(PMU1, 1, 10, 0, 10, 0, 0.2);
```

Where:

`Instr_id` = PMU1
`chan` = 1 (channel 1)
`VSrcRange` = 10 V
`Vrange_type` = Auto (0)
`Vrange` = 10 V (value ignored because V-measure autorange is set)
`Irange_type` = Auto (0)
`Irange` = 200 mA (value ignored because I-measure autorange is set)

pulse_remove Removes a channel from the test

Purpose

This function removes a pulse channel from the test.

Pulsers

Models 4220-PGU (VPU) and 4225-PMU

Pulse mode

Standard and Segment ARB

Format

```
int pulse_remove(INSTR_ID instr_id, int chan, double voltage,
unsigned long state);
```

`instr_id`	Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.
`chan`	Channel number: 1 or 2.
`voltage`	Voltage to output when removing a channel.
`state`	Output relay state: PULSE OUTPUT OFF or 0 = Open (disconnected) PULSE OUTPUT ON or 1 = Close (connected)

Remarks

Use this function to remove a channel from a test. It is useful when there needs to be one less channel for a pulse test that already exists. The `pulse_remove` function has two behaviors, depending on when it is used.

Use the `voltage` and `state` parameters to remove a channel from a test that is running. Use the `voltage` parameter to set the output voltage. For example, you may want to set the output voltage to zero (0) when removing the channel. Use the `state` parameter to connect or disconnect the channel. The output relay for the PMU is shown in Figure 16-2.

When removing a channel from a test that is not running, the `voltage` and `state` parameters are ignored.

Example The following function removes channel 1 for PMU2, sets the voltage to 0 V, and opens the output relay:

```
pulse_remove(PMU2, 1, 0, 0);
```

pulse_sample_rate Sets the sample rate

Purpose This function sets the measurement sample rate.

Pulsers Model 4225-PMU

Pulse mode Standard and Segment ARB

Format

```
int pulse_sample_rate(INSTR_ID instr_id, double Sample_rate);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`Sample_rate`	Sample rate: 200E6, 100E6, 50E6, 40E6, 33E6, 29E6, ... 1E3

Remarks Use this card-based function to set the measurement sample rate. The sample rate is the number of measurements (per second) that are performed by the PMU. The sample rate can be set from 200E6 to 200E6/n, where n = 1 to 200,000. The minimum sampling rate is 1E3 samples per second. The sample rate is a fixed rate (not adjustable within a test). For multi-card tests, set all cards to the same sample rate.

If a requested sample rate does not match an available rate, the next higher rate will be used. For example, if 90E6 samples per second is sent, the sampling rate will set to 100E6 samples per second (200E6/2).

Example The following function sets the sampling rate of the PMU to 100E6 samples per second:

```
pulse_sample_rate(PMU1, 100E6);
```

pulse_source_timing Sets pulse source timing

Purpose This function sets the pulse period, pulse width, rise time, fall time, and delay time.

Pulsers Models 4220-PGU (VPU) and 4225-PMU

Pulse mode Standard

Format

```
int pulse_source_timing(INSTR_ID instr_id, int chan, double
period, double delay, double width, double rise, double fall);
```

`instr_id`	Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.

`chan`	Channel number: 1 or 2.
`period`	Pulse period (in seconds) for both channels.
`delay`	Delay time (in seconds) for the selected channel.
`width`	Pulse width (in seconds for the selected channel.
`rise`	Rise time (in seconds) for the selected channel.
`fall`	Fall time (in seconds) for the selected channel.

Remarks

Use this function to set the timing parameters for the test. Pulse width, rise time, fall time, and delay are individually set for the selected channel. The pulse period setting applies to both channels. See Pulse parameter definitions for more information on these pulse parameters.

This function returns errors if there is an invalid setting or combination of settings.The rise time of a pulse cannot be longer than the pulse width. The minimum time allowed for parameters width, rise, and fall is 20 ns. The minimum value for delay is 0 ns. When setting timing for a sample (waveform capture), setting the delay to a small value allows the PMU to better capture the rising edge of the pulse. This value is sample rate dependent, but for the 200 MSa/s rate, a pulse delay of 20 ns to 100 ns will allow the rising edge of the pulse to be captured.

Another internally enforced limit is the minimum off time. This is calculated as:

$$\text{minimum off time} = \text{period} - \text{delay} - \text{width} - 0.5 \times (\text{rise} + \text{fall})$$

The minimum off time may not be less than 40 ns. To see the whole pulse transition to high when capturing waveform data, use a small non-zero value like 10 ns for `pulse_delay`.

When a source timing parameter is already set to step or sweep, the step or sweep parameter overrides the timing parameter set by this function. For details, see pulse_step_linear and pulse_sweep_linear.

For example, if the `SWEEP_PERIOD_SP` parameter type is selected for the `pulse_sweep_linear` function, the period values for the sweep will override the period setting for this function.

Example

This function sets the following pulse source timing settings for the PMU:

```
pulse_source_timing(PMU1, 1, 0.02, 0.005 0.01, 0.001, 0.001);
```

Where:

`instr_id` = PMU1
`chan` = 1
`period` = 0.02 (20 ms)
`delay` = 0.005 (5 ms)
`width` = 0.01 (10 ms)
`rise` = 0.001 (1 ms)
`fall` = 0.001 (1 ms)

pulse_step_linear Configures pulse stepping type
pulse_sweep_linear Configures pulse sweeping type

Purpose

These functions configure the pulse stepping type and pulse sweeping type.

Pulsers

Models 4220-PGU (VPU) and 4225-PMU

Pulse mode Standard

Format

```
int pulse_step_linear(INSTR_ID instr_id, int chan, int StepType,
double start, double stop, double step);

int pulse_sweep_linear(INSTR_ID instr_id, int chan int
SweepType, double start, double stop, double step);
```

instr_id	Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.	
chan	Pulse generator channel: 1 or 2.	
StepType/ SweepType	PULSE_AMPLITUDE_SP	Sweeps pulse voltage amplitude.
	PULSE_BASE_SP	Sweeps base voltage level.
	PULSE_DC_SP	Sweeps DC voltage level.
	PULSE_PERIOD_SP	Sweeps pulse period.
	PULSE_RISE_SP	Sweeps pulse rise time.
	PULSE_FALL_SP	Sweeps pulse fall time.
	PULSE_WIDTH_SP	Sweeps FWHM (full-width half-maximum) pulse width.
	PULSE_DUAL_BASE_SP	Dual sweeps base voltage level
	PULSE_DUAL_AMPLITUDE_SP	Dual sweeps pulse voltage amplitude
	PULSE_DUAL_DC_SP	Dual sweeps DC voltage level
start	Initial value for stepping/sweeping.	
stop	Final value for stepping/sweeping.	
step	Step size for stepping/sweeping.	

Remarks Use the pulse_step_linear function to configure stepping. Use the pulse_sweep_linear function to configure sweeping.

The relationship between a step function and a sweep function for SMUs is illustrated in Figure 6-153. The step/sweep relationship for pulsing is similar (see Figure 16-32). While a terminal of a device is at a pulse step, a pulse sweep is performed on another terminal. A pulse_step_linear function cannot be used by itself. At least one PMU channel in a test must be a valid pulse_sweep_linear function call. The last three sweep types are for pulse dual sweeps (see "Dual Sweep Option" on page 16-39).

Use the start, stop, and step parameters to configure stepping/sweeping. In addition, ensure that all pulse parameters are set before calling the pulse_sweep_linear or pulse_step_linear function. For example, when performing a pulse amplitude sweep (PULSE_AMPLITUDE_SP), use pulse_vlow to set the base voltage.

Amplitude and base level:

The pulse generator can step/sweep amplitude (with base level fixed) or step/sweep base level (with amplitude fixed). SweepType examples:

PULSE_AMPLITUDE_SP (stepping or sweeping): Start = 1 V, stop = 5 V, step = 1 V

Voltage amplitudes for pulse output sequence: 1 V, 2 V, 3 V, 4 V, and 5 V.

Note: Use the pulse_vlow function to set the base level voltage.

PULSE_BASE_SP (stepping or sweeping): Start = 5 V, stop = 1 V, step = -1 V

Voltage base levels for pulse output sequence: 5 V, 4 V, 3 V, 2 V, and 1 V.

Note: Use the pulse_vhigh function to set the amplitude voltage.

DC voltage level:

The pulse generator can step/sweep a DC level. `SweepType` example:

`PULSE_DC_SP` (stepping or sweeping): Start = 1 V, stop = 5 V, step = 1 V

DC voltage output sequence: 1 V, 2 V, 3 V, 4 V, and 5 V.

Pulse period:

The pulse period is the time interval between the start of the rising transition edge of consecutive output pulses (see Figure 11-27). `SweepType` example:

`PULSE_PERIOD_SP` (stepping or sweeping): Start = 0.01 s, stop = 0.05 s, step = 0.01 s

Pulse periods for output sequence: 0.01 s, 0.02 s, 0.03 s, 0.04 s, and 0.05 s.

Pulse rise time and fall time:

Pulse rise time is the transition time (in seconds) from pulse low to pulse high. Pulse fall time is the transition time from pulse high to pulse low (see Figure 11-32). `SweepType` examples:

`PULSE_RISE_SP` (stepping or sweeping): Start = 0.001 s, stop = 0.005 s, step = 0.001 s

Rise times for pulse output sequence: 0.001 s, 0.002 s, 0.003 s, 0.004 s, and 0.005 s.

`PULSE_FALL_SP` (stepping or sweeping): Start = 0.001 s, stop = 0.005 s, step = 0.001 s

Fall times for pulse output sequence: 0.001 s, 0.002 s, 0.003 s, 0.004 s, and 0.005 s.

Pulse width:

The width of a pulse (in seconds) is measured at full-width half-maximum (FWHM) as shown in Figure 11-28. `SweepType` example:

`PULSE_WIDTH_SP` (stepping or sweeping): Start = 0.01 s, stop = 0.05 s, step = 0.01 s

Pulse widths for pulse output sequence: 0.01 s, 0.02 s, 0.03 s, 0.04 s, and 0.05 s.

Dual Sweep:

The dual sweep allows for a voltage level sweep that goes up and down based on the voltage start stop and step. For example, a voltage amplitude sweep from 0 V to 4 V in 1 V steps. A single sweep (`PULSE_AMPLITUDE_SP`) would output 5 points: 0 V, 1 V, 2 V, 3 V, 4 V. A dual sweep version (`PULSE_DUAL_AMPLITUDE_SP`) outputs 10 points: 0 V, 1 V, 2 V, 3 V, 4 V, 4 V, 3 V, 2 V, 1 V, 0 V. See Figure 16-37 for a diagram of this example.

Example

The following function configures channel 1 of the PMU to perform an amplitude sweep from 1 V to 5 V in 1 V steps:

```
pulse_sweep_linear(PMU1, 1, PULSE_AMPLITUDE_SP, 1, 5, 1);
```

pulse_train Configures a pulse train

Purpose

This function configures the pulse generator to output a pulse train using fixed voltage values.

Pulsers

Models 4220-PGU (VPU) and 4225-PMU

Pulse mode

Standard

Format

```
int pulse_train(INSTR_ID instr_id, int chan, double Vbase,
double Vamplitude);
```

`instr_id`	Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.
`chan`	Pulse generator channel: 1 or 2.
`Vbase`	Voltage level for pulse base level.
`Vamplitude`	Voltage level for pulse amplitude.

Remarks The configured pulse train will not change for the selected channel, but any sweep or step timing changes will affect the timing parameters of the train. For details on timing, see pulse_step_linear and pulse_sweep_linear. A `pulse_train` function cannot be used by itself in a test. At least one PMU channel in a test must be a valid `pulse_sweep_linear` function call.

Example The following function configures channel 1 of the PMU to output a 0 to 5 V pulse train:

```
pulse_train(PMU1, 1, 0, 5);
```

rpm_config Configures the Model 4225-RPM

Purpose This function sends switching commands to the Model 4225-RPM.

Pulsers Model 4225-PMU with the Model 4225-RPM

Pulse mode Standard (two-level pulsing), Segment ARB, and full arb

Format

```
int rpm_config(INSTR_ID instr_id, long chan, long modifier, long
value);
```

`instr_id`	Instrument ID: PMU1, PMU2, and so on.
`chan`	Pulse generator channel: 1 or 2.
`modifier`	Parameter to modify: `KI_RPM_PATHWAY`.
`value`	Value to set `modifier`: `KI_RPM_PULSE` or 0: Selects pulsing (Model 4225-RPM) `KI_RPM_CV_2W` or 1: Selects 2-wire CVU (Model 4200-CVU) `KI_RPM_CV_4W` or 2: Selects 4-wire CVU (Model 4200-CVU) `KI_RPM_SMU` or 3: Selects SMU (Model 4200-SMU)

Remarks The Model 4225-RPM includes input connections for the Models 4200-CVU and Model 4200-SMU. Use this function to control switching inside the RPM to connect the PMU, CVU, or SMU to the output.

When using the PMU with the RPM, `rpm_config` must be called to connect the pulse source to the RPM output. Note that if there is no RPM connected to the PMU channel, the `rpm_config` command will not cause an error. The RPM connection is cleared by the `clrcon` command.

Example The following function sets channel 1 of the RPM for pulsing:

```
rpm_config(PMU1, 1, KI_RPM_PATHWAY, KI_RPM_PULSE);
```

See also

seg_arb_sequence Defines a Segment ARB pulse-measure sequence

Purpose This function defines the parameters for a Segment ARB pulse-measure sequence.

Pulsers Models 4220-PGU (VPU) and 4225-PMU

Pulse mode Segment ARB

Format

```
int seg_arb_sequence(INSTR_ID inst_id, long chan, long SeqNum,
long NumSegments, double *StartV, double *StopV, double *Time,
long *Trig, long *SSR, long *MeasType, double *MeasStart, double
*MeasStop);
```

`instr_id`	Instrument ID: VPU1, VPU2, PMU1, PMU2, and so on.
`chan`	The pulse generator channel: 1 or 2.
`SeqNum`	Sequence ID number (1 to 512, per channel) to uniquely identify this sequence.
`NumSegments`	Total number of segments in this sequence.
`StartV`	An array of start voltage levels.
`StopV`	An array of stop voltage levels.
`Time`	An array of segment time durations (in seconds with 10 ns resolution, 20 ns minimum).
`Trig`	An array of trigger values: 0 (trigger low) or 1 (trigger high).
`SSR`	An array of values to control the high endurance output relay: 0 (open) or 1 (closed).
`MeasType`	PMU only: An array of measure types: 0 = No measurements for this segment 1 = Spot mean discrete (see Figure 8-117) 2 = Waveform discrete (see Figure 8-120 and Figure 8-121) 3 = Spot mean average 4 = Waveform average
`MeasStart`	PMU only: An array of start measurement times (in seconds, with 10 ns resolution). A zero (0) second setting sets measure to start at the beginning of the segment.
`MeasStop`	PMU only: An array of stop measurement times (in seconds, with 10 ns resolution). This is the elapsed time, within the segment, when the measurement stops. The Model 4220-PGU does not have pulse-measure capability. When this function for the PGU is called, the parameter values for `MeasType`, `MeasStart`, and `MeasStop` are ignored.

Remarks For the PMU, use this function to configure each channel to output its own unique Segment ARB® waveform and perform measurements. For the PGU, use this function to configure each channel to output its own unique Segment ARB waveform.

A Segment ARB sequence is made up of user-defined segments (up to 2048 per channel). Each sequence can have a unique start voltage, stop voltage, time interval,

output trigger level (TTL high or low), output relay state (open or closed), pulse measurement type (for PMU), measurement start time (for PMU), and measurement stop time (for PMU).

A defined sequence is uniquely identified by its specified channel number and sequence ID number. This function defines the sequences, or building blocks, that are typically used for a BTI (bias temperature instability) test.

A sequence is defined as three or more segments with seamless (no voltage differences) voltage transitions. Seamless means that the voltage level for the last point in a segment must equal the voltage level for the first point of the next segment. Note that all segment transitions must be seamless. The minimum time per sequence is 20 ns.

One or more defined sequences are then combined into a Segment ARB® waveform using the seg_arb_waveform function. All sequence transitions must also be seamless. Seamless means that the voltage level for the last point in a sequence must equal the voltage level on the first point of the next sequence. Figure 8-125 shows an example of a waveform that consists of three sequences.

NOTE *See Section 5 of the User's Manual for details on using KPulse to output Segment ARB waveforms.*

Figure 8-124 shows an example of a Segment ARB sequence defined by the `seg_arb_sequence` function. Spot mean discrete measurements are performed on segments two and four.

Figure 8-124
Segment ARB sequence example

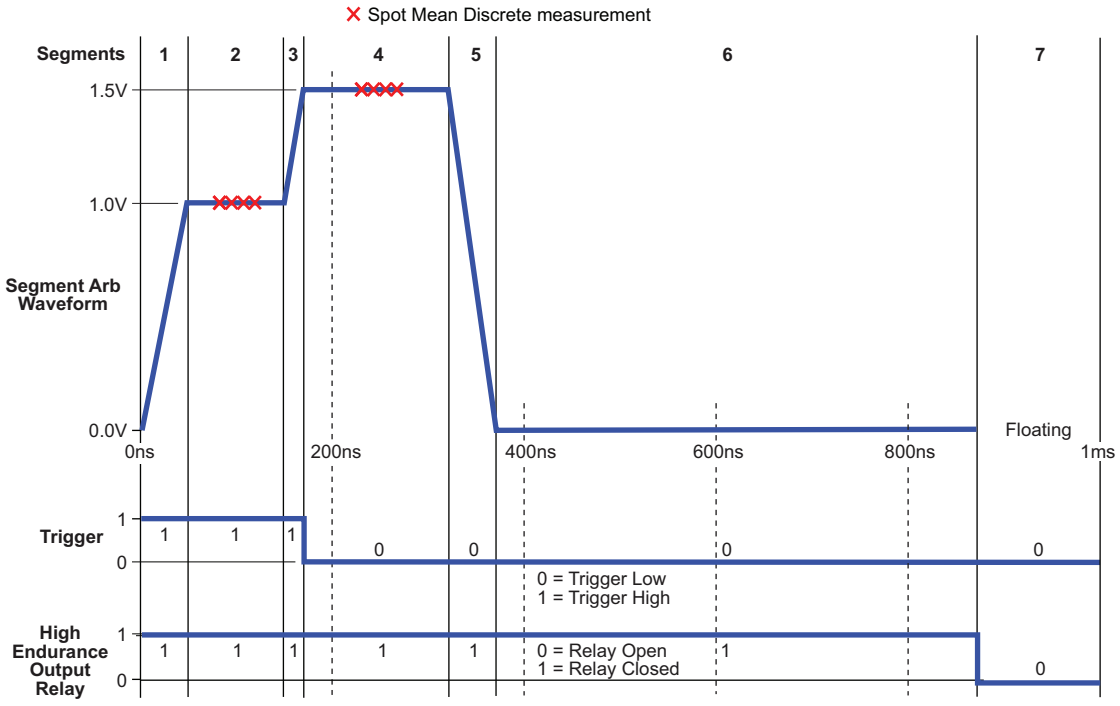


Table 8-22 lists the `seg_arb_sequence` parameter arrays for the Segment ARB sequence shown in Figure 8-124.

Table 8-22
Parameter arrays for the seg_arb_sequence example

Parameter	Array Name	Value						
SegNum	Seg_Num	1	2	3	4	5	6	7
StartV	Start_Volt	0 V	1 V	1 V	1.5 V	1.5 V	0 V	0 V
StopV	Stop_Volt	1 V	1 V	1.5 V	1.5 V	0 V	0 V	0 V
Time	Time_Interval	50e-9 s	100e-9 s	20e-9 s	150e-9 s	50e-9 s	500e-9 s	130e-9 s
Trig	Trigger_Level	1 (high)	1 (high)	1 (high)	0 (low)	0 (low)	0 (low)	0 (low)
SSR	Output_Relay	1 (closed)	1 (closed)	1 (closed)	1 (closed)	1 (closed)	1 (closed)	0 (open)
MeasType	Meas_Type	0 (none)	1 (spot mean)	0 (none)	1 (spot mean)	0 (none)	0 (none)	0 (none)
MeasStart	Meas_Start	0 s	25e-9 s	0 s	50e-9 s	0 s	0 s	0 s
MeasStop	Meas_Stop	0 s	75e-9 s	0 s	100e-9 s	0 s	0 s	0 s

Example

The following function defines the Segment ARB sequence shown in Figure 8-124:

```
seg_arb_sequence(PMU1, 1, 1, 7, Start_Volt, Stop_Volt,
Time_Interval, Trig_Level, Output_Relay, Meas_Type, Meas_Start,
Meas_Stop);
```

Arrays for the `seg_arb_sequence` function:

```
double Start_Volt[7] = {0, 1, 1, 1.5, 1.5, 0, 0};
double Stop_Volt[7] = {1, 1, 1.5, 1.5, 0, 0, 0};
double Time_Interval[7] = {50e-9, 100e-9, 20e-9, 150e-9, 50e-9,
500e-9, 130e-9};
int Trig_Level[7] = {1, 1, 1, 0, 0, 0, 0};
int Output_Relay[7] = {1, 1, 1, 1, 1, 1, 0};
int Meas_Type[7] = {0, 1, 0, 1, 0, 0, 0};
double Meas_Start[7] = {0, 25e-9, 0, 50e-9, 0, 0, 0};
double Meas_Stop[7] = {0, 75e-9, 0, 100e-9, 0, 0, 0};
```

seg_arb_waveform Creates a Segment ARB pulse-measure waveform

Purpose

This function creates a voltage segment waveform.

Pulsers

Models 4220-PGU (VPU) and 4225-PMU

Pulse mode

Segment ARB

Format

```
int seg_arb_waveform(INSTR_ID inst_id, long chan, long NumSeq,
long *Seq, double *SeqLoopCount;
```

`instr_id` Instrument ID: VPU1, VPU2, and so on.

`chan` The pulse generator channel: 1 or 2.

`NumSeq`	Total number of sequences in waveform definition (512 maximum).
`Seq`	An array of sequences using the sequence number ID (see `SeqNum` parameter for the seg_arb_sequence function).
`SeqLoopCount`	An array of loop values (number of times to output a sequence).Loop value range is 1 to 1E12.

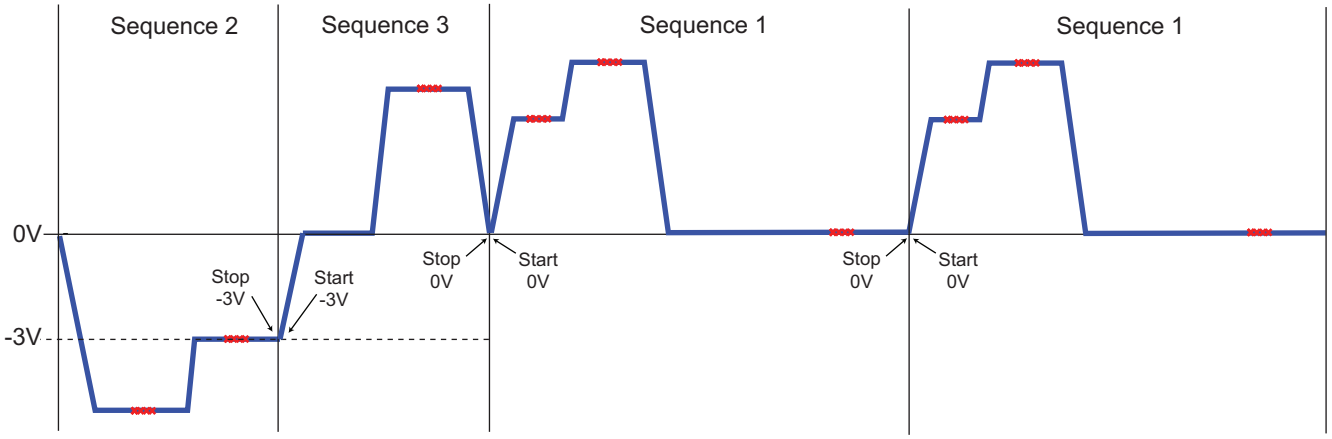
Remarks

Use this function to create a voltage segment waveform from the sequences defined by the seg_arb_sequence function. The `NumSeq` parameter defines the number of sequences that make up the waveform. The `Seq` parameter is an array that indicates the identification (ID) number for each sequence in the waveform. The sequence ID numbers are set by the seg_arb_sequence function.

You can use this function to configure a waveform that repeats one or more of its sequences with the `SeqLoopCount` parameter.

All sequence transitions must be seamless. Seamless means that the voltage level for the last point in a sequence must equal the voltage level on the first point of the next sequence. Figure 8-125 shows an example of a three-sequence waveform that uses looping (Sequence 1 is repeated). Notice that the start and stop voltage values between sequences are the same, making it seamless.

Figure 8-125
Three-sequence waveform (with looping)



Example

The following function configures channel 1 of the PMU for a single three-sequence Segment ARB® waveform (see Figure 8-125). This example assumes that the three sequences shown in Figure 8-125 have already been defined by the seg_arb_sequence function.

```
seg_arb_waveform(PMU1, 1, 3, Seq_Num, Seq_Loop_Count);
```

Arrays for the waveform:

```
int Seq_Num[3] = {2, 3, 1};
double Seq_Loop_Count[3] = {1, 1, 2};
```

setmode Set the number of iterations for load line effect compensation (LLEC)

Purpose

This function sets the number of iterations for LLEC for the PMU.

Pulsers

Model 4225-PMU

NOTE *The following information pertains specifically to the Model 4225-PMU. For details on using the setmode function for other instruments, see setmode for SMU and system settings and setmode for Model 4200-CVU settings.*

Pulse mode Standard and Segment ARB

Format

```
int setmode(INSTR_ID inst_id, long modifier, double value);
```

`instr_id` Instrument ID of the pulse generator: PMU1, PMU2, and so on.

`modifier` Parameters to set maximum iterations tolerance windows:

`KI_PXU_LLC_MAX_ITERATIONS`	Maximum iterations
`KI_PXU_CH1_LLC_TOLERANCE`	Acceptance window (Ch 1)
`KI_PXU_CH2_LLC_TOLERANCE`	Acceptance window (Ch 2)
`KI_PXU_CH1_LLC_OFFSET`	LLEC Offset (Ch 1)
`KI_PXU_CH2_LLC_OFFSET`	LLEC Offset (Ch 2)

`value` Values to set maximum iterations and tolerance window:

Max iterations: 1 to 1000; typical setting is 20 to 30 iterations
Acceptance window: 0.0001 (0.01 percent) to 1 (100 percent); typical range is 0.001 (0.1 percent) to 0.01 (1 percent), with 0.003 (0.3 percent) the typical value. Offset value: Value is in volts

Remarks Use this function to set the number of iterations for load line effect compensation (LLEC). LLEC is an algorithm, running on each PMU in the test, that adjusts the output of the PMU to respond to the device-under-test resistance and reach the desired (programmed) output value at the DUT. This algorithm is not deterministic, meaning it cannot be guaranteed to reach the desired target value. Therefore, there are controls to fine tune the LLEC performance.

When enabled, the algorithm performs a number of iterations to determine the appropriate output voltage. The pulse_meas_sm and pulse_meas_wfm functions enable or disable LLEC. See Load line effect compensation (LLEC) for more information on LLEC.

LLEC is configured by setting the number of maximum iterations that will be performed and setting an acceptance window for one or both PMU channels. LLEC will continue until either the output voltage to the DUT falls within the acceptance window or until the maximum number of iterations are performed.

Use `KI_PXU_LLC_MAX_ITERATIONS` to set the maximum number of iterations (integer from 1 to 1000) to be performed. LLEC only performs the iterations that are needed to determine the appropriate voltage setting to provide the desired level at the DUT. The remaining iterations are not performed.

Use `KI_PXU_CHx_LLC_OFFSET` to set the offset of the tolerance window

Use `KI_PXU_CHx_LLC_TOLERANCE` to set the gain of the tolerance window (in percentage of desired signal level).

The LLEC tolerance window:

LLEC window = LLC_TOLERANCE * Desired Voltage + LLC_OFFSET

LLEC is satisfied when:

Measured voltage < Desired voltage +/- LLEC Window

For example, assume the programmed pulse output is 1 V and the acceptance window is set to 0.1 (10 percent) and offset to 10 mV. LLEC will perform iterations until the output voltage falls within the 0.9 V to 1.1 V window

Setting a smaller tolerance will result in voltage steps that are much closer to the desired voltage steps sizes, but at the expense of longer test times.

NOTE *When selecting and configuring an LLEC iteration method, remember that testing speed is affected by the maximum number of iterations as well as the tolerance window. Choosing a high maximum number of iterations and a tight tolerance will result in much longer test times.*

Example The following function sets the LLEC for channel 1 of the PMU for a one percent acceptance window:

```
setmode(PMU1, KI_PXU_CH1_LLC_TOLERANCE, 0.01);
```

LPT Library Status and Error codes

There are two types of codes reported in KITE, in the message area shown in Figure 6-1. The positive values are status or updates, and the negative values are errors and warnings. This section does not include error codes for UTMs, such as 4200-PIV-A or 4200-FLASH. See the Model 4200 Applications Manual for the UTM-based application error codes.

Each error code number is associated with a brief text explanation. However, many of the error texts are customized with specific information, such as a particular SMU or ID number. See the Key for an explanation of the type of customized data.

In addition to error codes, some conditions may prevent a valid measurement condition. In these cases, the reported measurement value itself will report a condition that is usually a large number with an exponent of 10^{22} or 10^{23}. See Table 8-25 for the conditions associated with these large numbers.

Table 8-23
LTP Library status and error codes

Key	Explanation
%d	Signed decimal number may be a parameter index or GPIB address
%g	Double value
%i	Signed decimal number
%s	String, such as “SMU1” or other test resource
%u	Unsigned integer
%04x	Hexadecimal number, 4 places
%08x	Hexadecimal number, 8 places

Table 8-24
Code status or error titles

Code	Status or error titles
2802 – 2807	RPM: Invalid Configuration Requested
2801	RPM: Returned ID Error Response

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
2800	RPM: Command Response Timeout
2702	PMU: Temperature Within Normal Range
2701	PMU: High Temperature Limit Exceeded
1905	PMU: Measure Program Error
1904	PMU: Source Program Error
1900	PMU: DA Communication Timeout
400 – 402	PMU: Invalid Attributes in SW Command
100	LPTLib is executing function %s on instrument ID %d.
55	%s is no longer in thermal shutdown.
54	%s VXIBus device busy (command ID %04x). Timed out after %g seconds.
53	%s VXIbus transaction recovered after %u timeouts.
52	%s VXIbus transaction (command ID %04x) timed out after %g seconds.
51	Interlock reset.
50	Interlock tripped.
40	%s
24	Config %d-%d complete for %s (%d).
23	Config %d-%d starting for %s (%d).
22	Binding %s (%d) to driver %s.
21	Loading driver %s.
20	Preloading model code %08x (%s).
15	Executor started.
14	%s channel closed.
13	%s channel starting.
12	TAPI services shutting down.
11	Starting TAPI services.
9	System configuration complete.
8	System configuration starting.
4	System initialization complete.
1	The call was successful (no error).
0	The call was successful (no error).
-4	Too many instruments in configuration file %s.
-5	Memory allocation failure.
-6	Memory allocation error during configuration with configuration file %s.
-20	Command not executed because a previous error was encountered.
-21	Tester is in a fatal error state.
-22	Fatal condition detected while in testing state.
-23	Execution aborted by user.
-24	Too many arguments.
-25	%s is unavailable because it is in use by another test station.
-40	%s.
-87	Can not load library %s.
-88	Invalid configuration file %s.
-89	Duplicate IDs
-90	Duplicate instrument addresses in configuration file %s.
-91	Duplicate instrument slots in configuration file %s.
-93	Unrecognized/missing interface for %s in configuration file %s.
-94	Unrecognized/missing PCI slot number for %s in configuration file %s.

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
-95	Unrecognized/missing GPIB address for %s in configuration file %s.
-96	GPIB Address out of range for %s was %i in configuration file %s.
-97	PCI slot number out of range for %s was %i in configuration file %s.
-98	Error attempting to load driver for model %s in configuration file %s.
-99	Unrecognized/missing instrument ID in configuration file %s.
-100	Invalid connection count, number of connections passed was %d.
-100	Invalid connection count, number of connections passed was %d.
-101	Argument #%d is not a pin in the current configuration.
-102	Multiple connections on %s.
-103	Dangerous connection using %s.
-104	Unrecognized instrument or terminal not connected to matrix, argument #%d.
-105	No pathway assigned to argument #%d.
-106	Path %d previously allocated.
-107	Not enough pathways to complete connection.
-108	Argument #%d is not defined by configuration.
-109	Illegal test station: %d.
-110	A ground connection MUST be made.
-111	Instrument low connection MUST be made.
-113	There are no switching instruments in the system configuration.
-114	Illegal connection.
-115	Operation not allowed on a connected pin: %d.
-116	No physical bias path from %s to %s.
-117	Connection cannot be made because a required bus is in use.
-118	Cannot switch to high current mode while sources are active.
-119	Pin %d in use.
-120	Illegal connection between %s and GNDU.
-121	Too many calls were made to trigXX.
-122	Illegal value for parameter #%d.
-124	Sweep/Scan measure table overflow.
-126	Insufficient user RAM for dynamic allocation.
-129	Timer not enabled.
-137	Invalid value for modifier.
-138	Too many points specified in array.
-139	An error was encountered while accessing the file %s.
-140	%s unavailable while slaved to %s.
-141	Timestamp not available because no measurement was made.
-142	Cannot bind, instruments are incompatible.
-143	Cannot bind, services unavailable or in use.
-152	Function not supported by %s (%d).
-153	Instrument with ID %d is not in the current configuration.
-154	Unknown instrument name %s.
-155	Unknown instrument ID %i.
-158	VXI device in slot %d failed selftest (mfr ID: %04x, model number: %04x).
-159	VME device with logical address %d is either non-VXI or non-functional.
-160	Measurement cannot be performed because the source is not operational.
-161	Instrument in slot %d has non-functional dual-port RAM.
-164	VXI device in slot %d statically addressed at reserved address %d.

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
-165	Service not supported by %s (%d).
-166	Instrument with model code %08x is not recognized.
-167	Invalid instrument attribute %s.
-169	Instrument %s is not in the current configuration.
-190	Ill-formed connection.
-191	Mode conflict.
-192	Instrument sense connection MUST be made.
-200	Force value too big for highest range %g.
-202	I-limit value %g too small for specified range.
-203	I-limit value %g too large for specified range.
-204	I-range value %g too large for specified range.
-206	V-limit value %g too large for specified range.
-207	V-range value %g too large for specified range.
-213	Value too big for range selection, %g.
-218	Safe operating area for device exceeded.
-221	Thermal shutdown has occurred on device %s.
-224	Limit value %g too large for specified range.
-230	V-limit value %g too small for specified range.
-231	Range too small for force value.
-233	Cannot force when not connected.
-235	C-range value %g too large for specified range.
-236	G-range value %g too large for specified range.
-237	No bias source.
-238	VMTR not allocated to make the measurement.
-239	Timeout occurred attempting measurement.
-240	Power Limited to 20 W. Check voltage and current range settings.
-250	IEEE-488 time out during data transfer for addr %d.
-252	No IEEE-488 interface in configuration.
-253	IEEE-488 secondary address %d invalid for device.
-254	IEEE-488 invalid primary address: %d.
-255	IEEE-488 receive buffer overflow for address %d.
-261	No SMU found, kelvin connection test not performed.
-262	SRU not responding.
-263	DMM not connected to SRU.
-264	GPIB communication problem.
-265	SRU not mechanically calibrated.
-266	Invalid SRU command.
-267	SRU hardware problem.
-268	SRU kelvin connection problem.
-269	SRU general error.
-270	Floating point divide by zero.
-271	Floating point log of zero or negative number.
-272	Floating point square root of negative number.
-273	Floating point pwr of negative number.
-280	Label #%d not defined.
-281	Label #%d redefined.
-282	Invalid label ID #%d.

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
-301	PCI ID read back on send error, slot.
-455	Protocol version mismatch.
-510	No command byte available (read) or SRQ not asserted.
-511	CAC conflict.
-512	Not CAC.
-513	Not SAC.
-514	IFC abort.
-515	GPIB timed out.
-516	Invalid function number.
-517	TCT timeout.
-518	No listeners on bus.
-519	Driver problem.
-520	Bad slot number.
-521	No listen address.
-522	No talk address.
-523	IBUP Software configuration error.
-524	No utility function.
-550	EEPROM checksum error in %s: %s.
-551	EEPROM read error in %s: %s.
-552	EEPROM write error in %s: %s.
-553	%s returned unexpected error code %d.
-601	System software internal error; contact the factory.
-602	Module load error: %s.
-603	Module format error: %s.
-604	Module not found: %s.
-610	Could not start %s.
-611	Network error.
-612	Protocol error.
-620	Driver load error. Could not load %s.
-621	Driver configuration function not found. Driver is %s.
-640	%s serial number %s failed diagnostic test %d.
-641	%s serial number %s failed diagnostic test %d with a fatal fault.
-650	Request to open unknown channel type %08x.
-660	Invalid group ID %d.
-661	Invalid test ID %d.
-662	Ill-formed list.
-663	Executor is busy.
-664	Invalid unit ID %d.
-701	Error configuring serial port %s.
-702	Error opening serial port %s.
-703	Call kspcfg before using kspsnd or ksprcv.
-704	Error reading serial port.
-705	Timeout reading serial port.
-706	Terminator not received before read buffer filled.
-707	Error closing serial port %s.
-801	Exception code %d reported from VPU in slot %d, channel %d.
-802	VPU in slot %d has reached thermal limit.

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
-803	Start and stop values for defined segmented arb violate minimum slew rate.
-804	Function not valid in the present pulse mode.
-805	Too many points specified in array.
-806	Not enough points specified in array.
-807	Function not supported by 4200-VPU.
-808	Solid state relay control values ignored for 4200-VPU.
-809	Time Per Point must be between %g and %g.
810	Attempts to control VPU trigger output are ignored by the 4200-VPU.
-811	Measure range not valid for %s.
-812	WARNING: Sequence %d, segment %d. Cannot measure with PGUs/VPUs.
-820	PMU segment start value %gV at index %d does not match previous segment stop value of %gV.
-821	PMU segment stop time (%g) greater than segment duration (%g)
-822	PMU sequence error for entry %d. Start value %gV does not match previous stop value of %gV.
-823	Start and stop window was specified for PMU segment %d, but no measurement type was set.
-824	Measurement type was specified for PMU segment %d, but start and stop window is invalid.
-825	%s set to post to column %s. Cannot fetch data that was registered as real-time.
-826	Cannot execute PMU test. No channels defined.
-827	Invalid pulse timing parameters in PMU Pulse IV test.
-828	Maximum number of segments per PMU channel exceeded (%d).
-829	The sum of base and amplitude voltages (%gV) exceeds maximum (%gV) for present range.
-830	Pulse waveform configuration exceeded output limits. Increase pulse period or reduce amplitude or total time of pulsing.
831	Maximum number of samples per channel (%d) exceeded for PMU%d-CH%d.
-832	Pulse slew rate is too low. Increase pulse amplitude or reduce pulse rise and fall time.
-833	Invalid trigger source for PIV test.
-834	Invalid pulse timing parameters.
-835	Using the specified sample rate of %g samples/s, the time (%g) for sequence %d is too short for a measurement.
-836	WARNING: Sequence %d, segment %d is attempting to measure while solid state relay is open. Disabling measurement.
-837	No RPM connected to channel %d of PMU in slot %d.
-838	Timing parameters specify a pulse that is too short for a measurement using %g samples/s.
-839	Timing parameters contain measurement segment(s) that are too short to measure using %g samples/s.
-840	SSR cannot be opened when using RPM ranges. Please change SSR array to enable relay or select PMU measure range.
-841	WARNING: SSR is open on segment immediately preceding sequence %d. Measurement will be invalid for 25 us while relay settles.
-842	This test has has exceeded the system power limit by %g watts.
-843	Step size of %g is not evenly divisible by 10 ns.
-844	Invalid combination of start %g1, stop %g2 and step %g3.
-845	No pulse sweeper was configured - Test will not run.
-846	Maximum Source Voltage Reached: Requested voltage across DUT resistance exceeds maximum voltage available.
-847	Output was not configured - Test will not run.

Table 8-24 (continued)
Code status or error titles

Code	Status or error titles
-848	Sweep step count mismatch for the sweeping channels. All sweeping channels must have same # of steps.
-849	ILimit command is not supported for RPM in slot %d, channel %d.
-850	Sample Rate mismatch. All channels in test must have the sample rate.
-851	Invalid PxU stepper/sweeper configuration.
-900	Environment variable KI_PRB_CONFIG is not set. The prober drivers will be inaccessible.
-901	Environment variable KI_PRB_CONFIG contains an invalid path. The prober drivers will be inaccessible.
-902	Prober configuration file not found. File was %s. The prober drivers will be inaccessible.
-903	Unable to copy the prober configuration %s to %s. The prober driver many not be unable.
-10000 – -20000	User Module (UTM) error codes. Refer to user module description (help) for details.

Table 8-25
Large number reported readings and explanations

Measurement Value	Condition
1.0000E+22 or 10.0000E+21	SMU is in range compliance (see Types of compliance in Section 3), where the reading is at the maximum of a fixed range.
5.0000E+22	SMU measurement at the maximum SMU voltage or current.
7.0000E+22	SMU in real compliance (see Types of compliance in Section 3).
1.0000E+23	Measurement aborted or not able to be performed. For example, if using LPT command to make a measurement, but the SMU output was not enabled, then this measurement value will be reported.

LPTLib and KITE interaction via UTMs

ITMs and UTMs are typically independent. However, an ITM and a UTM are not independent if 1) the UTM occurs before an ITM in the project plan, and 2) the UTM configures a switch matrix. Under these conditions, the following occur:

- KITE assumes that the ITM depends on the UTM-created switch configuration.
- Therefore, KITE maintains the UTM-created switch configuration during execution of the ITM.

Refer to Table 8-26.

Table 8-26
KITE actions affected by ITM and UTM sequence

Test sequence in the project plan	KITE action
A UTM precedes an ITM	Before the ITM executes, the `devint` function initializes all devices, *except* for the switch matrix (the switch configuration is preserved to run the subsequent ITM).
A UTM precedes a UTM	No initialization operations occur.
An ITM precedes an ITM	No LPTLib calls occur.
An ITM precedes a UTM	Before the UTM executes, the `devint` function initializes all devices, *including* the switch matrix.

Cross-platform LPTLib compatibility

The LPT Library (LPTLib) is included with the Model 4200-SCS to provide an application programming interface (API) for controlling instrumentation and accessing I/O. LPTLib is available on the Keithley Instruments Models S400 and S600 series parametric test systems for the same purpose. Therefore, user libraries can be moved between the Keithley Instruments Model 4200-SCS, Model S400, and Model S600 test systems. In most cases, user libraries can be moved from one system to another with little or no modification to the user library source code. Simply recompile and build the library on the target platform, and the library is ready for use. However, in some cases it is necessary to modify the user library to address platform-specific differences in LPTLib functions.

Table 8-27 indicates the compatibility of each LPTLib function across all LPTLib-based Keithley Instruments test systems, using the following symbols:

- A dash (**-**) indicates that the corresponding LPTLib function is not supported on the indicated Keithley Instruments test system.
- An **X** indicates that the function is supported on the indicated Keithley Instruments test system.
- A superscripted numeral (**1**, **2**, or **3**) next to a dash (**-**) or **X** indicates that the corresponding LPTLib function behaves differently on each platform. This number refers to a footnote that describes the significant platform differences.

Unless otherwise indicated, unsupported LPTLib functions in Table 8-27 cannot be used in a KULT user module. Use of an unsupported function causes a build error of the form:

```
<module_name>.obj: error LNK2001: unresolved external symbol _<function_name>
```

Use of an unsupported function may also cause a compilation error of the form:

```
<path\module_name>.c: warning C4013: _<function_name> undefined;...
```

For more detailed information regarding moving Model 4200-SCS user libraries to and from the Models S400 and S600, refer to Moving user libraries: Model 4200-SCS to Model S400 and Moving user libraries: Model 4200-SCS to a Model S600/S630 later in this section.

Table 8-27
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
Instrument	devclr	X	X	X
	devint	X	X	X
	setvims	-	X	-
	setvmtr	-	X	-
	setimtr	-	X	-
Matrix	addcon	X	X	X
	conpin	X	X	X
	conpth	X	X	X
	clrcon	X	X	X
	delcon	X	X	X
	floatpin	-	-	X
Ranging	atten	-	X	-
	lorangei	X	X	X
	lorangev	X	X	X
	lorangec	-	X	X
	lorangeg	-	X	X
	rangei	X	X	X
	rangev	X	X	X
	rangec	- [1]	X	X
	rangeg	- [1]	X	X
	setauto	X	X	X
Sourcing	forcei	X	X	X
	forcev	X	X	X
	limiti	X	X	X
	limitv	X	X	X
	outebl	-	X	-
	pulsev	X	X	-
	pulsei	X	X	-
Measuring	avgi	X	X	X
	avgv	X	X	X
	avgc	- [1]	X	X
	avgg	- [1]	X	X
	fltoff	-	X	-
	flton	-	X	-

Table 8-27 (continued)
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
	intgi	X	X	X
	intgv	X	X	X
	intgc	-	X	X
	intgg	-	X	X
	measi	X	X	X
	measv	X	X	X
	measc	-[1]	X	X
	measg	-[1]	X	X
	measf	-	X	-
	meast	X	X	X
	setac	-	X	-
	setdc	-	X	-
	setfilter	-	X	X
	setgate	-	X	-
	settrig	-	X	-
	ssmeasi	-	X	X
	ssmeasv	-	X	X
	ssmeasc	-	X	-
	ssmeasg	-	X	-
	nslope	-	X	-
	pslope	-	X	-
	zchoff	-	X	-
	zchon	-	X	-
Combination	asweepi	X	X	X
	asweepv	X	X	X
	bmeasi	X	X	-
	bmeasv	X	X	-
	bmeasc	-	X	-
	bmeasg	-	X	-
	bsweepv	X	X	X
	bsweepi	X	X	X
	clrtrg	X	X	X
	clrscn	X	X	X
	mpulse	X	X	-
	rtfary	X	X	X
	savgi	X	X	X

Table 8-27 (continued)
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
	savgv	X	X	X
	savgc	-	X	X
	savgg	-	X	X
	scnmeas	X	-	X
	searchi	X	X	X
	searchv	X	X	X
	sintgi	X	X	X
	sintgv	X	X	X
	sintgc	-	-	X
	sintgg	-	-	X
	smeasi	X	X	X
	smeasv	X	X	X
	smeasc	- [1]	X	X
	smeasg	- [1]	X	X
	smeast	X	X	X
	sweepi	X	X	X
	sweepv	X	X	X
	trigcomp	X	-	X
	trigig	X	X	X
	trigvg	X	X	X
	trigcg	-	X	X
	triggg	-	X	X
	trigrg	-	X	-
	trigfg	-	X	-
	trigtg	X	X	X
Combination	trigil	X	X	X
	trigvl	X	X	X
	trigcl	-	X	X
	triggl	-	X	X
	trigrl	-	X	-
	trigfl	-	X	-
	trigtl	X	X	X
Timing	adelay	X	X	X
	delay	X	X	X
	rdelay	X	X	X
	enable	X	X	X

Table 8-27 (continued)
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
	imeast	X	X	X
	disable	X	X	X
	retmrstats	-	X	X
Pulse	arb_array	X	-	-
	arb_file	X	-	-
	pg2_init	X	-	-
	pulse_burst_count	X	-	-
	pulse_current_limit	X	-	-
	pulse_dc_output	X	-	-
	pulse_delay	X	-	-
	pulse_fall	X	-	-
	pulse_halt	X	-	-
	pulse_init	X	-	-
	pulse_load	X	-	-
	pulse_output	X	-	-
	pulse_output_mode	X	-	-
	pulse_period	X	-	-
	pulse_range	X	-	-
	pulse_rise	X	-	-
	pulse_ssrc	X	-	-
	pulse_trig	X	-	-
	pulse_trig_output	X	-	-
	pulse_trig_polarity	X	-	-
	pulse_trig_source	X	-	-
	pulse_vhigh	X	-	-
	pulse_vlow	X	-	-
	pulse_width	X	-	-
	seg_arb_define	X	-	-
	seg_arb_file	X	-	-
Execution & Synchronization	execut	X[2]	X	X
	inshld	X	X	X
	kthtimo	-	X	-
	rexcut	-	X	-
	syncmode	-	-	X
	xrf_buffer_size	-	X	-

Table 8-27 (continued)
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
Arithmetic	kfpabs	X[3]	X	X[3]
	kfpadd	X[3]	X	X[3]
	kfpdiv	X[3]	X	X[3]
	kfpexp	X[3]	X	X[3]
	kfplog	X[3]	X	X[3]
	kfpmul	X[3]	X	X[3]
	kfpneg	X[3]	X	X[3]
	kfppwr	X[3]	X	X[3]
	kfpsqrt	X[3]	X	X[3]
	kfpsub	X[3]	X	X[3]
Parallel I/O	pior	-	X	X
	piorb	-	X	X
	piow	-	X	X
	piowait	-	X	X
	piowb	-	X	X
GPIB	ibup	-	X	X
	kibcmd	X	X	X
	kibdefclr	X	X	X
	kibdefint	X	X	X
	kibdefdelete	X	-	-
	kibrcv	X	X	X
	kibsnd	X	X	X
	kibspl	X	X	X
	kibsplw	X	X	X
RS-232	kspcfg	X	-	-
	kspsnd	X	-	-
	ksprcv	X	-	-
	kspdefclr	X	-	-
	kspdefdelete	X	-	-
	kspdefint	X	-	-
Branch	klpbeq	-	X	-
	klpbge	-	X	-
	klpbgt	-	X	-
	klpble	-	X	-
	klpblt	-	X	-
	klpbne	-	X	-

Table 8-27 (continued)
LPTLib function compatibility

Group	Function / module	4200-SCS	S400	S600
	klpbra	-	X	-
	klplbl	-	X	-
General	beep	-	-	X
	compclr	-	-	X
	getinstid	X	-	-
	getinstname	X	-	-
	getinstattr	X	-	-
	getstatus	X	X	X
	insbind	-	-	X
	insinfo	-	-	X
	setmode	X	X	X
	refctrl	-	-	X
	prbsel	-	X	-
	tstdsl	X	X	X
	tstsel	X	X	X
Error handling	display_lpt_error	-	X	-
	extract_lpt_error	-	X	-
	getlpterr	X	X	X
	log_lpt_error	-	X	-
	kthvmerror	-	X	-

Notes:
1. LPTLib functions to facilitate capacitance measurements are not directly supported on the Model 4200-SCS. However, user libraries for controlling the Keithley Instruments Model 590 CV Analyzer and the Hewlett Packard Model 4980 LCR Meter are provided with the Model 4200-SCS. Refer to Capacitance-meter support differences later in this section for more information.
2. `execut()` simply calls `devint()` on the Model 4200-SCS. It does not execute the program.
3. Provided for legacy user library compatibility purposes only. Usage of this LPTLib command is not recommended. Use the ANSI C-language equivalent.

S400/S600 functions not supported by the Model 4200-SCS

The following list summarizes the functions not supported by the Model 4200-SCS:

- **Database calls**:

```
PutLot, PutWafer, PutSite, PutParam,
PutParamList, EndLot, EndWafer, EndSite,
GetLot, GetWafer, GetSite, GetParam,
GetParamList, GetLotData, LogLot, LotWaf,
LogSit, LogPtr, LogPta, MatchParam2Limit,
FileExist, LotExist, GetStartTime,
DeleteLot, DeleteWafer, DeleteSite,
DeleteParam, DeleteLimitCode, DeleteLimit,
GetComment, PutComment, GetLimitCode,
GetLimit, PutLimit, GetLotStats,
```

GetWaferStats, FindLot, FindData, AddNewLimit, CreateNewLimit, FindFirstLimit, FindLastLimit, FindNextLimit, FindPrevLimit, InsertNewLimit, RemoveLimit

- **KUI (Keithley user interface) calls:** GetProgramArgs, InitUI, InputMsgDlg, LotDlg, OkCancelAbortMsgDlg, OkCancelDlg, OkMsgDlg, QuitUI, ScrollMsgDlg, ScrollMsgDlgMsg, StatusDlg, UpdateModelessDlgs, UpdateStatusDlg, VerifyAbort, WfrIdsDlg, WfrIdDlg, YesNoAbortMsgDlg, YesNoCancelMsgDlg
- **KWF (Keithley wafer file) calls:** AddNewSubsite, CreateNewSubsite, FindFirstSubsite, FindNextSubsite, FindSubsiteId, readWDF
- **PARLIB (PARameter LIBrary) calls:** Bchk, Beta1, Beta2, Beta2a, Beta3a, Bice, Bice1, Bice2, Bvcbo, Bvcbo1, Bvceo, Bvceo1, Bvceo2, Bvces, Bvces1, Bvebo, Ev, Ibic1, Ibic2, Ibic3, Icbo, Iceo, Ices, Iebo, Is1, Is2, Pbice, Pbice1, Pbice2, Picib, Prb, Pvcic, Pvcicr, Rb, Rcsat, Re, Rev, Vbes, Vbibic, Vcesat, Vcic, Vcicr, Bkdn, Cap, Capg, Con, Evalcj, Fimv, Fndcj, Fvmi, Leak, Meascp, Meascs, Pcp, Pcs, Psimv, Psvmi, Rcont, Res, Res2, Res4, Resv, Rsq, Rvdp, Simv, Svmi, Tox, Vf, Bvdss, Bvdss1, Deltl1, Deltw1, Gamma1, Gd, Id1, Idsat, Idvsvd, Idvsvg, Isubmx, Pidvd, Pidvg, Pimax, Pvtvbs, Vg2, Vg2a, Vgsat, Vt14, Vt14s, Vtati, Vtext, Vtext2, Vtext3, Vtvbs, Gm, Idss, Imax, Rsd, Rsg, Vp, Vp1

Moving user libraries: Model 4200-SCS to Model S400

This section describes the issues involved with moving an Model S400UX C-language function to the Model 4200-SCS and moving a Model 4200-SCS function to the Model S400UX. It is important to note that this section does not cover the porting of code from the Model S400 VAX to the Model 4200-SCS, because FORTRAN-to-C code conversions are beyond the scope of this document.

Header files

When you move functions, generally you need not move header files if they are covered by the ANSI C standard (for example, `stdio.h`, `time.h`, and so on). However, three important exceptions follow:

- The first exception is in the case of *absolute path names*. If the Model S400 UNIX path name is hard-coded in the source code as follows:

  ```
  /pathname1/pathname2/HeaderFileName.h
  ```

  then in the Model 4200-SCS, this path name must be corrected to reflect the header file's location on the embedded computer hard drive. For example:

  ```
  c:\pathname1\pathname2\HeaderFileName.h
  ```

- The second exception is in the case of non-standard or UNIX-specific header files. For example, the Model S400 header file `/usr/include/sys/asynch.h` has no Windows® equivalent. You must locate a suitable replacement for such a header file when using an associated function in the Model 4200-SCS.

- The third exception is in the case of Model S400 header files related to the Keithley Instruments KDF (Keithley data files) database. There is no equivalent to the KDF database on the Model 4200-SCS. Therefore, when using Model S400 code in a Model 4200-SCS, remove any reference to the KDF database.

Instrument hardware differences

On the Model S400, the source/measure units were referred to as VIMS. Therefore, when using Model S400 code in the Model 4200-SCS, you must replace any instance of the string `VIMS` with the string `SMU`.

The terminals that are referred to as GPTs (general-purpose terminals) on the Model S400 are referred to as GPIs (general-purpose instruments) on the Model 4200-SCS. Therefore, you must replace the string `GPT` in the Model S400 code with the string `GPI` in the Model 4200-SCS code.

The following instruments that were supported on the Model S400 are either not supported on the Model 4200-SCS or are used in a vastly different way on the Model 4200-SCS:

- **FMTR**: The Model 4200-SCS does not support a frequency counter.
- **PSRC**: The Model 4200-SCS does not support the power source. In many instances, if the current needed is 1 A or less, you may be able to use an SMU as a replacement.
- **VMTR**: There is no separate voltmeter for the Model 4200-SCS. The Model S400 used separate voltmeters, such as the KI2001 and Model 192/196, to provide low-voltage measurement capabilities. On the Model 4200-SCS, an SMU provides equivalent low-voltage measurement capabilities.
- **IMTR**: There is no separate current meter for the Model 4200-SCS. The Model S400 uses separate current meters, such as the Model 617 and 9162-PAU (VME), to provide picoampere-level measurement capabilities. The Model 4200-SCS SMUs are capable of picoampere-level and sub-picoampere-level measurement.
- **VSRC**: The Model 4200-SCS does not currently support high-voltage instruments.
- **CMTR**: The Model 4200-SCS supports the KI590 and HP4980 capacitance meters. However, the Model 4200-SCS supports these capacitance meters through user libraries that are not call-compatible with Model S400 user libraries or Model S400 LPT functions (on the Model S400, the CMTR was supported through VME-level software drivers). For more information about the differences, refer to Capacitance-meter support differences later in this section.

As a rule of thumb, VMTRs, PSRCs, and IMTRs of the Model S400 can be replaced with SMUs on the Model 4200-SCS. The functionality of the ki590ulib and hp4980ulib user libraries is equivalent to the functionality of the Model S400 LPT capacitance-meter functions.

Instrument range differences

Table 8-28 shows the range differences between the Model 4200-SCS SMUs and the Model S400 VIMS.

Table 8-28
Range differences: Model 4200-SCS SMUs and Model S400 VIMS

System	Instrument	Voltage ranges	Current ranges
S400	VIMS	200 V, 40 V, 4 V, 0.4 V	200 mA, 20 mA, 2 mA, 200 µA, 20 µA, 2 µA, 200 nA, 20 nA
	IMTR (PAU)	Not applicable	20 mA, 2 mA, 200 µA, 20 µA, 2 µA, 200 nA, 20 nA, 2 nA, 200 pA
	Microvoltmeter	200 mV, 2 V, 20 V, 200 V	Not applicable
	MIVS (medium current voltage source, PSRC)	20 V	1 A, 10 A
4200-SCS	SMU	200 mV, 2 V, 20 V, 200 V	1 pA, 10 pA, 100 pA, 1 nA, 10 nA, 100 nA, 1 µA, 10 µA, 100 µA, 1 mA, 10 mA, 100 mA, 1 A

Capacitance-meter support differences

The Model 4200-SCS and Model S400 systems support capacitance meters in very different ways. When measuring capacitance and conductance on the Model S400, you generally used standard capacitance/conductance-measurement functions such as `measc`, `measg`, `intgc`, `intgg`, `rangec`, `rangeg`, `avgc`, `avgg`, `savgc`, `savgg`, `sintgc`, and `sintgg`. The Model 4200-SCS uses no equivalent commands. Instead, the Model 4200-SCS supports capacitance meters through Keithley Instruments-supplied user-library user modules (or, more specifically, through KITE UTMs that are connected to Keithley Instruments-supplied user modules). Table 8-29 provides guidance in substituting Model 4200-SCS user modules for Model S400 functions.

Table 8-29
Capacitance-meter support differences: Model 4200-SCS SMUs and Model S400 VIMS

S400 CMTR function PostDataDouble (PMU)	**Equivalent Model 4200-SCS user-library user module**
`measc, measg`	Cmeas590 or Cmeas4284
`bmeasc, bmeasg, intgc, intgc, sintgc, sintgg, ssmease, ssmeasg, trigcg, triggg, trigcl, triggl`	n/a
`smeasc, smeasg`	Cvsweep590 or Cvsweep4284
`rangec, rangeg`	Cmeas590 or Cmeas4284
`forcev(CMTRx...)`	Cmeas590 or Cmeas4284
`avgc, avgg`	Use Cmeas590 or Cmeas4284 and mathematically average

Absence of the PARLIB library on the Model 4200-SCS

The Model 4200-SCS does not support the `PARLIB`. However, the `PARLIB` was supplied in source code form on the Model S400 and S600. Therefore, you can use the guidelines described in this section to move PARLIB code from the Model S400/S600 to the Model 4200-SCS.

Absence of the KDF database on the Model 4200-SCS

There is no equivalent to the Keithley Instruments KDF database on the Model 4200-SCS. You must eliminate the database calls when porting to the Model 4200-SCS.

Parameter differences

Many Keithley Instruments LPT functions on the Model S400 required `float` input arguments. On the Model 4200-SCS, these same calls require `double` input arguments. For example, if your Model S400 function contained the following:

```
float range = 1e-6;
       *
       *
       *
rangei(VIMS1, range);
```

this code must be changed to the following to work on the Model 4200-SCS:

```
double range = 1e-6;
      *
      *
      *
rangei(SMU1, range);
```

LPT execution differences

There are fundamental differences between the ways the Model 4200-SCS and Model S400 execute LPT commands. These differences are listed below:

- **Command synchronization**:
  On the Model S400, LPT commands were first downloaded to the VME instruments in packets. These packets were executed completely and independently of the applications processor (for example: the Sun workstation) and only returned results when an `execut`, `inshld`, or `rexcut` command was encountered. This is called results synchronization. By contrast, on the Model 4200-SCS, all commands are executed sequentially by the internal CPU and results are returned as soon as they are available.

  When moving code from the Model S400 to the Model 4200-SCS, no changes to your code are necessary to accommodate this fundamental difference. However, when moving code from the Model 4200-SCS to the Model S400, you must add the `execut`, `inshld`, or `rexcut` commands at the appropriate location in your source code. For example, if your Model 4200-SCS code performs a mathematical operation on a measured result, such as the following:

  ```
  measv(SMU1, &voltage);
  resistance = voltage/current;
  ```

  you would need to add a call, after the `measv` call, either to `inshld` (which holds all instruments in their current state) or to `execut` (which executes all previous LPT commands and returns all instruments to their default states). In the latter case, your code would be modified to read as follows:

  ```
  measv(SMU1, &voltage);
  execut();
  resistance = voltage/current;
  ```

- **Instrument clearing**:
  On the Model S400, the `execut` command caused all previously issued LPT commands to execute. Once all LPT commands executed, all instruments were cleared and set to their default states, and all matrix connections were cleared. However, on the Model 4200-SCS, you may call `execut()`, which initiates a `devint()` call.

Moving user libraries: Model 4200-SCS to a Model S600/S630

The issues described above under Moving user libraries: Model 4200-SCS to Model S400, also apply to moving KULT libraries between the S600 and Model 4200-SCS, *with the following notable exceptions*:

- **Instrument names**:
  The source-measure units on the Model S600 and Model 4200-SCS have the same instrument IDs (for example, SMUx, where x = 1 to 8). You need not make any changes to your code to change the SMU instrument IDs.
- **Parameter differences**:
  The input parameter types for Keithley measurement functions are exactly the same on the Model S600 and the Model 4200-SCS.
- **Instrument differences**:
  On the Model S600, the general-purpose instrument terminals have instrument IDs called FOHMx (where x is 1 to 8). On the Model 4200-SCS, the equivalent terminals have instrument IDs called GPIx (where x is an integer). When moving C-language code between the platforms, you must make the appropriate substitutions or conditionalize your code with the `#ifdef` preprocessor statement.
- **SMU current range differences**:
  When equipped with a preamplifier, a Model 4200-SCS SMU has two additional ranges that do not exist on the Model S600: 10 pA and 1 pA. When moving source code to the Model S600, make sure that you make the appropriate changes to your source code.
- **SMU voltage ranges**:
  The Models 4200-SCS and S600 have identical voltage ranges.
- **LPT function differences**:
  Refer to Table 8-27 for the list of LPT functions supported by the Models 4200-SCS and S600. In cases where functions on one system are not supported on the other system, you must make appropriate changes to the source code when moving between the systems.
- **Command synchronization**:
  When moving code from the Model 4200-SCS to the Model S600, you generally need not make changes to the code to account for command synchronization.
  When moving code from the Model S600 to the Model 4200-SCS, be aware that the S600 has several synchronization modes that are not supported by the Model 4200-SCS. However, it is generally safe to comment out any code that changes the Model S600 synchronization method.

Section 9

Keithley External Control Interface (KXCI)

In this section:

Introduction

- **Remote control overview (GPIB, Ethernet)**: Provides an overview of KXCI operation.
- **KXCI control interface**: Describes how to use the KXCI user interface, which is used to control GPIB operation.
- **GPIB command set**: The command strings to control the source-measure operations of the Model 4200 are summarized in three tables.
- **GPIB command reference**: Provides detailed reference information on the command set.
- **Ethernet command reference:** Describes the command set that can be used for Ethernet communication.
- **SMU default settings**: Describes how to return the SMUs to their power-on default settings. Includes a table that lists the default settings.
- **Output data formats**: Describes the format of the data string that returns measured readings to the computer for both system mode and user mode.
- **Status byte and serial polling**: Describes how to use the status byte and serial polling to monitor operating conditions of the Model 4200-SCS.
- **Sample programs**: Provides three programs to demonstrate the following operations over the GPIB: VAR1 and VAR2 sweep operation, basic source-measure operation, and retrieving a data file.
- **GPIB error messages**: Provides a list of GPIB error messages and numbers.
- **Pulse generator and scope commands**: Provides reference information for the commands to control the pulse generator card and scope card.
- **KXCI CVU commands**: Describes the command set to control the CVU using KXCI in both user mode and system mode.
- **Calling KULT user libraries remotely**: Describes the set of KXCI commands available to make use of KULT user libraries from a remote interface.
- **KXCI Ethernet client driver**: Describes the supplied driver DLL used to control KXCI through the Ethernet.

Remote control overview (GPIB, Ethernet)

The Keithley External Control Interface (KXCI) allows you to use an external computer to remotely control the SMUs, pulse generator cards and scope card in the Model 4200-SCS over the general purpose instrument bus (GPIB) or Ethernet (as defined in KCON). When controlled by an external computer, the Model 4200-SCS functions like any other GPIB instrument.

The KXCI GPIB command sets for the SMUs and PG2s are similar to the command set used by the HP Model 4145B. This similarity allows many programs already developed for the HP Model 4145B run on the Model 4200-SCS.

GPIB standards

The GPIB is the IEEE-488 instrumentation data bus with hardware and programming standards originally adopted by the Institute of Electrical and Electronic Engineers (IEEE) in 1975. The Model 4200 conforms to these standards:

- IEEE-488.1-1987
- IEEE-488.2-1992

Communication connections

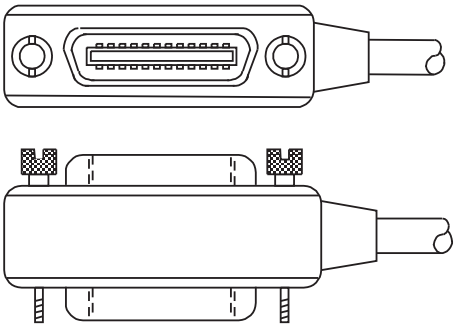
GPIB connections: To connect the Model 4200-SCS to the GPIB, use a cable equipped with standard IEEE-488 connectors as shown in Figure 9-1. Either end of this cable mates to the

IEEE-488 connector on the rear panel of the Model 4200-SCS (Figure 9-2). Connect the other end of the cable to the IEEE-488 connector on the computer.

The connectors on the cable are stackable to allow GPIB connection to other instruments. However, to avoid damage, do not stack more than three connectors on any one unit.

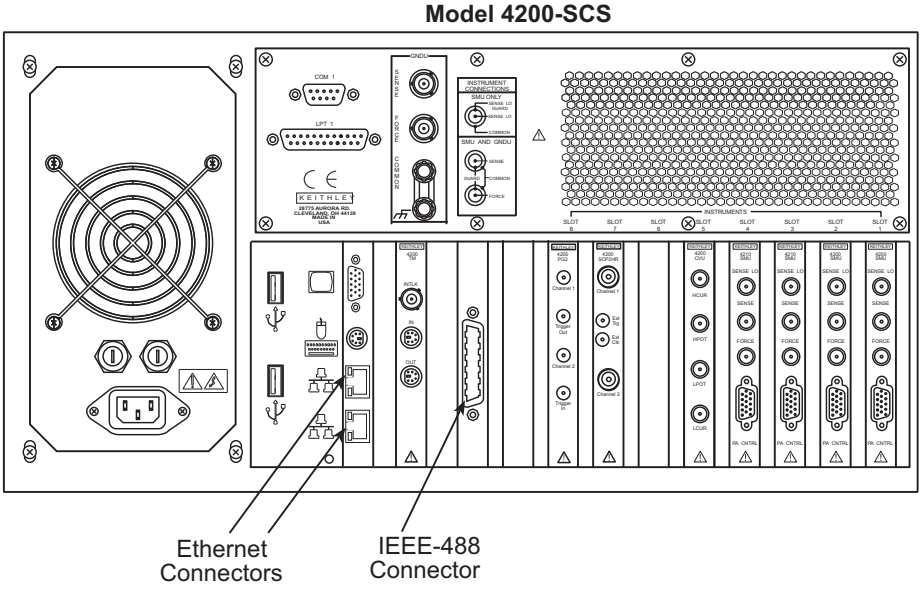
NOTE *To minimize interference caused by electromagnetic radiation, use shielded IEEE-488 cables such as the Keithley Instruments Models 7007-1 and 7007-2.*

Figure 9-1
IEEE-488 Cable connectors



Ethernet connections: Use a standard cable (CAT-5, RJ-45 terminated) to connect to the Model 4200-SCS (see Figure 9-2).

Figure 9-2
IEEE-488 and Ethernet connectors on Model 4200-SCS



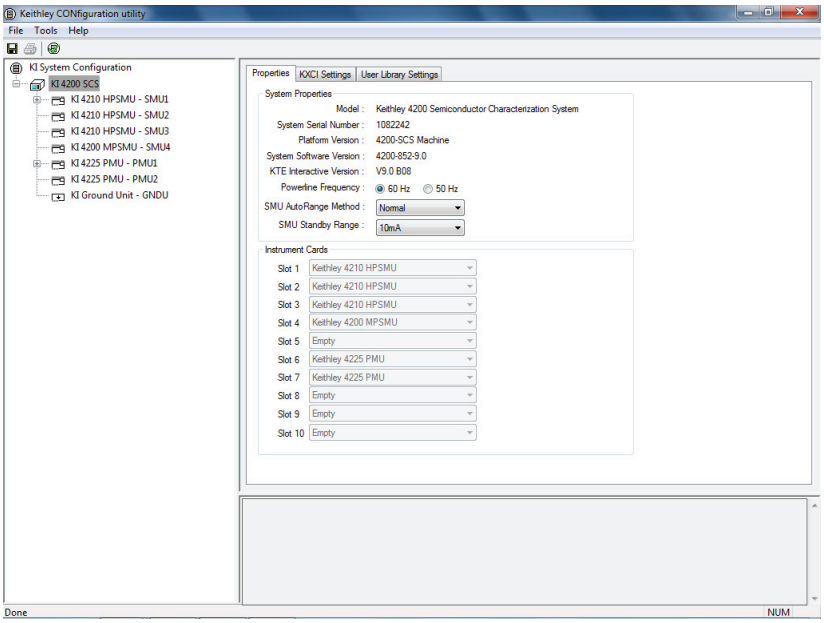
Setting remote control mode (GPIB versus Ethernet)

Setting remote control mode on the 4200 is done through KCON on the KXCI settings tab. By default the 4200 is setup for GPIB remote control.

Select the remote control mode on the Model 4200-SCS:

1. Open KCON and click the **KI 4200-SCS** system node (see Figure 9-3).

Figure 9-3
KI 4200 SCS properties tab

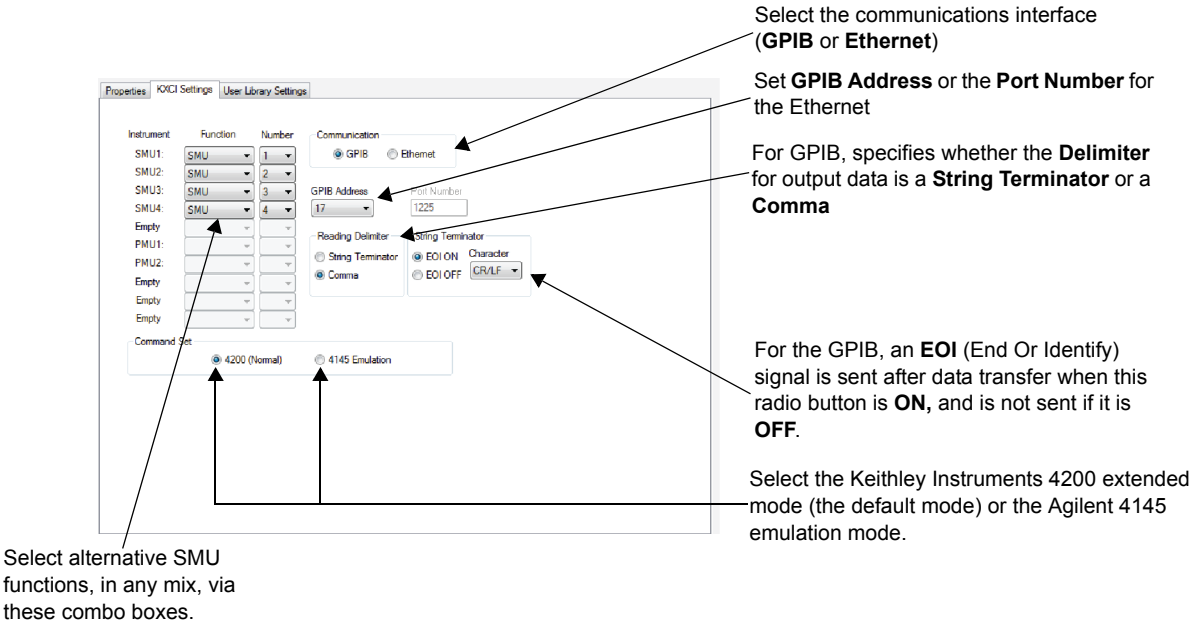


KXCI settings tab

The **KXCI Settings** tab stores the KXCI settings. KXCI allows the Model 4200-SCS to behave as a GPIB slave, or Ethernet slave (see Figure 9-4).

2. Click the KXCI settings tab.

Figure 9-4
KI 4200 SCS KXCI settings tab



KXCI facilitates using an external computer to remotely control the Model 4200-SCS over the GPIB bus and Ethernet. In many cases, test programs developed for use with an Agilent 4145B run without modification when they are used with a Model 4200-SCS running KXCI. For detailed

information regarding KXCI refer to the Reference Manual, Keithley External Control Interface (KXCI), Section 9.

The **Function**, **Communication**, **GPIB Address/Port Number**, **Delimiter**, **EOI**, and **Command Set** settings on the **KXCI Settings** tab are each described in the Reference Manual, KXCI Settings tab, page 7-16.

KXCI control interface

Starting KXCI and the GPIB command interpreter

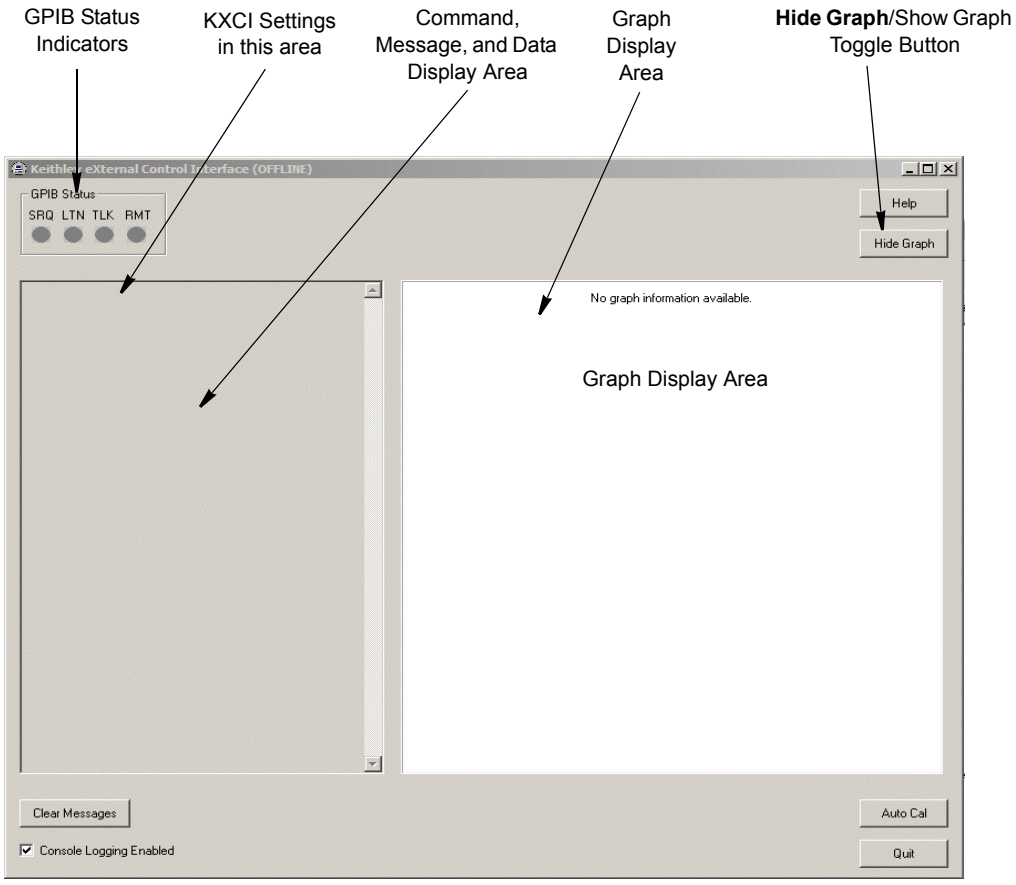
Start the KXCI program by one of the following methods:

- Double-click the **KXCI** icon on the desktop.
- Click **Start > Programs > Keithley > KXCI**.

The KXCI user interface appears and the command interpreter starts (see Figure 9-5).

NOTE *The **GPIB Status Indicators** only apply if GPIB communication is selected.*

Figure 9-5
KXCI user interface



The KXCI user interface and command interpreter controls the Model 4200-SCS over the GPIB or Ethernet, as defined in KCON. For detailed information regarding KXCI refer to the Reference Manual, Keithley External Control Interface (KXCI), Section 9.

NOTE *Keithley Interactive Test Environment (KITE) and KXCI are mutually exclusive applications and cannot be run at the same time.*

***KITE**: In this mode, the Model 4200-SCS is the controller and controls all internal and external instruments.*

***KXCI**: In this mode, the Model 4200-SCS is a slave to a controlling PC over GPIB or Ethernet.*

NOTE *For further information about controlling the Model 4200-SCS through Ethernet, refer to "Communication connections" on page 9-3 and "KXCI Ethernet client driver" on page 9-106.*

Understanding the KXCI user interface

KXCI settings

The Keithley CONFiguration (KCON) utility is used to configure KXCI. Use KCON to assign source-measure functions to the installed SMUs, and to select and configure communications (GPIB or Ethernet):

- GPIB: In KCON, set the GPIB address, select a delimiter, and enable or disable EOI.
- Ethernet: In KCON, set the IP address and the number of the port.

For details on KXCI settings, refer to Section 7.

NOTE *Before opening KCON to change the present KXCI configuration, you must first close KXCI.*

The presently selected communications interface (GPIB or Ethernet) and its settings are displayed in the *KCXI* console. The command and message area below the KXCI settings displays sent commands. KXCI error message, and numerical test results (refer to Using KXCI).

GPIB status indicators (GPIB communications only)

After KXCI is started, status is provided by four indicators located in the GPIB **Status** box on the interface. A green indicator signals the present status.

SRQ: Turns on when an error or operating condition occur.

LTN: When on, instrument is in the listener active state.

TLK: When on, instrument is in the talker active state.

RMT: When on, instrument is in the remote state.

Graph display

In response to optional graphics commands, the right side of the KXCI user interface displays a graph of the test results. Clicking the **Hide Graph** button hides the graph from view, thereby providing a larger display area for commands, error messages, and test results.

Understanding the log file

If the **Console Logging Enabled** check box is checked (lower right of the user interface), KXCI logs all GPIB commands to a file named `KXCILogfile.txt`. The text file can be opened after KXCI is closed. Note that whenever KXCI is opened, the log file clears.

The log file is accessed using the following directory:

C:\S4200\sys\KXCI\KXCILogfile.txt

The text file can be opened from Windows Explorer. After opening the **KXCI** folder, double-click the text file name to open it using Notepad. It can also be opened using any other text editor.

Using KXCI

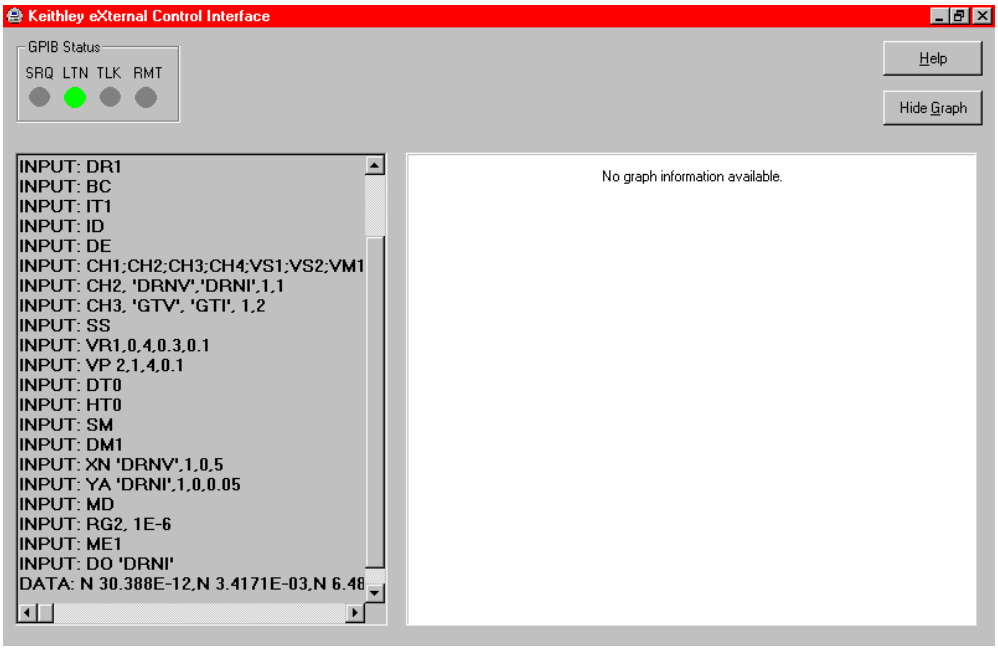
To start GPIB operation, start KXCI. The Model 4200-SCS is ready to accept GPIB commands immediately after you start KXCI (for command information, refer to GPIB command set, GPIB command reference, Ethernet command reference, and Pulse generator and scope commands).

Logging commands, errors, and test results

When you send GPIB commands, KXCI logs the commands, error messages, and test results as follows:

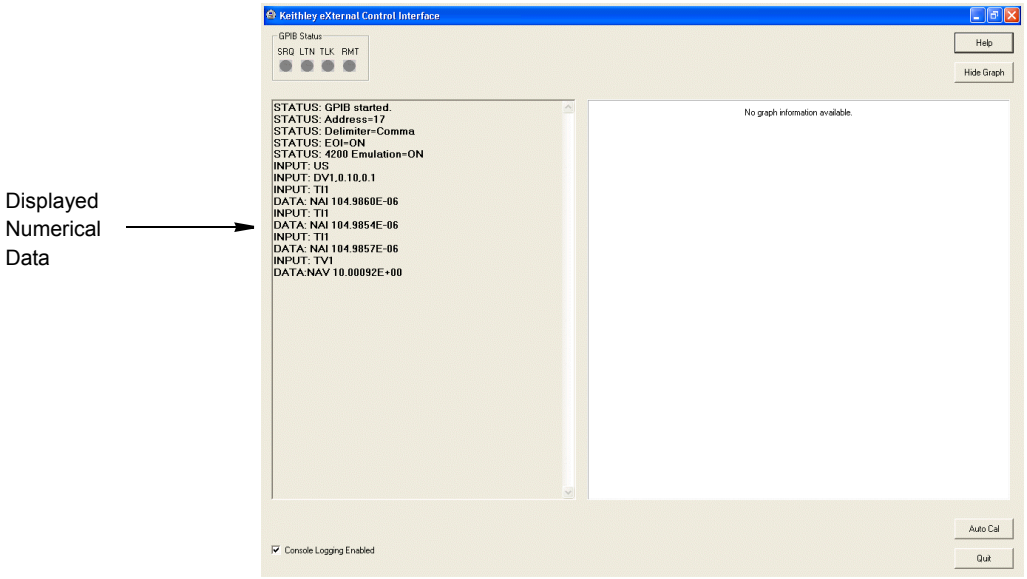
- When you send a command:
  - The left side of the user interface (the command and message display area) displays each command as it is received. See Figure 9-6.

Figure 9-6
Command display



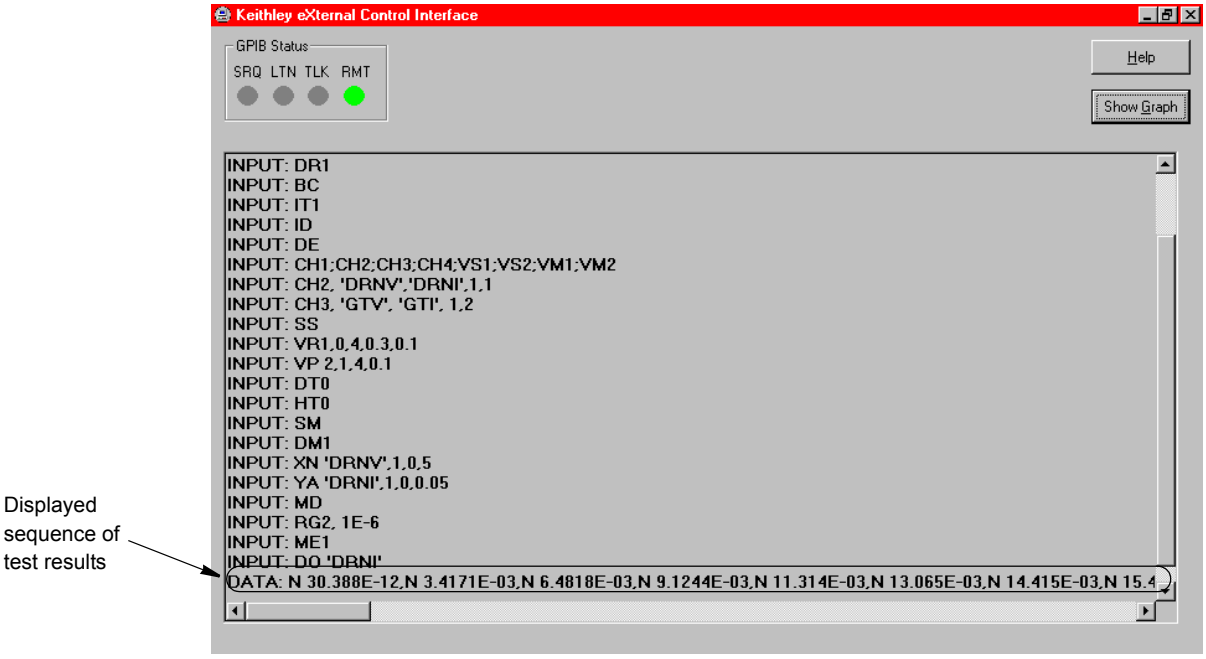
  - If the **Console Logging Enabled** check box is checked, KXCI also logs each command into the KXCI log file (C:\S4200\sys\KXCI\KXCILogfile.txt).
- The command and message display area displays error messages as they occur.
- The command and message display area also displays the numerical test results, both in the 4200 extended mode and 4145 emulation mode (refer to Section 7). See Figure 9-7 and Figure 9-8.

Figure 9-7
Test results in command and message display area



NOTE *Figure 9-8 shows the graph display hidden, using the **Hide Graph** button, to better display a long sequence of test results.*

Figure 9-8
Hidden graph area



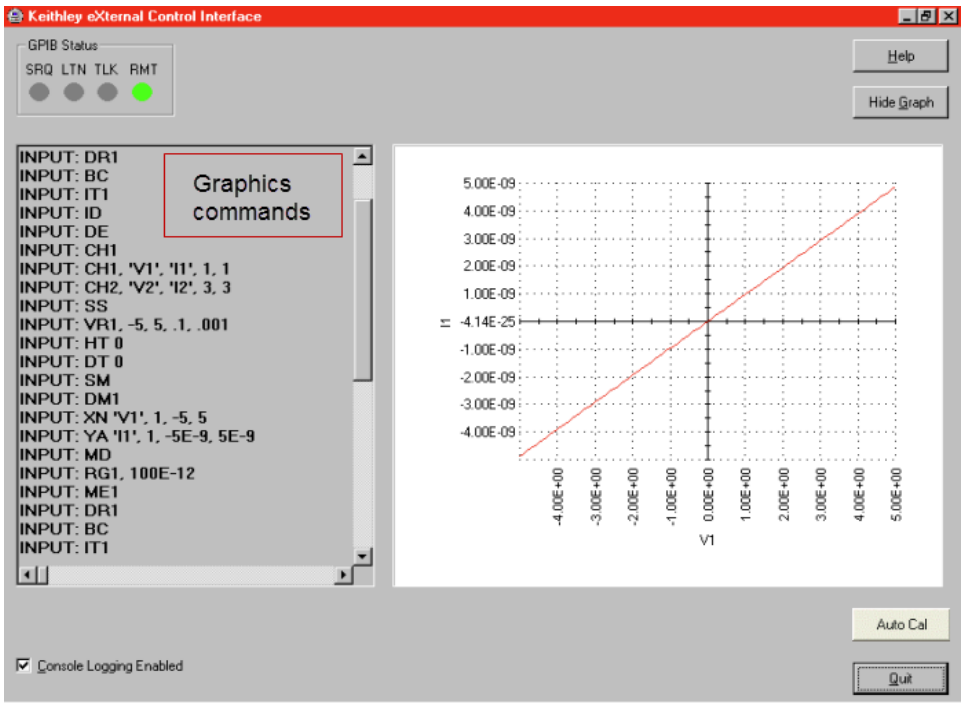
- If the sent commands include the needed graphics commands, the graph display area graphs the test data. Refer to the next subsection.

To stop GPIB operation, close KXCI by clicking the **Quit** button on the interface.

Graphing the test results

If you have sent the needed graphics commands, the DM1 command followed by the X axis and Y axis configuration commands, KXCI displays a graph of the generated data. See the example graph and highlighted graphics commands in Figure 9-9.

Figure 9-9
Data graph



NOTE *KXCI plots all Y1 axis curves in red and all Y2 axis curves in blue. You can temporarily hide the graph by clicking the* ***Hide Graph*** *button.*

GPIB command set

GPIB commands to control instrument operation are divided into four categories:

- **Page commands:** KXCI locates the GPIB instrument control routines in various pages that are similar to the HP Model 4145B command pages. Consequently, before sending an instrument control command string, you must send the appropriate page command. The page commands are summarized in Table 9-1.
- **System mode commands**: This comprehensive set of commands uses all of the source-measure capabilities of up to eight SMUs installed in the Model 4200-SCS. These commands are summarized in Table 9-2.
- **User mode commands**: This limited set of commands perform basic source-measure operation. These commands are summarized in Table 9-3.
- **Common commands**: These commands are valid in any operating mode. These commands are summarized in Table 9-4.
- **4200 extended mode-only commands**: These commands are specific to the 4200 extended mode (not usable in the 4145 emulation mode). They are summarized in Table 9-5.

NOTE *Detailed information on the command set is presented after the tables.*

Table 9-1
Page commands

Page command	Function
DE	Accesses SMU channel definition page.
SS	Accesses source setup page.
SM	Accesses measurement setup page.
MD	Accesses measurement control page.
US	Accesses user-mode page.
UL	Allows for use of direct user library module calls - See "Calling KULT user libraries remotely."

Table 9-2
System mode commands

Page[1]	Command	Function	Command String
DE	CH	SMU channel definition	CHA, 'BBBBBB', 'CCCCCC', D, E
			A = 1, 2, ... or n SMU channel number.[2] The largest permissible value of n equals the number of channels in the system (8 maximum).
			BBBBBB = Voltage name
			CCCCCC= Current name
			D = 1 Voltage source mode
			= 2 Current source mode
			= 3 Common (high connected to common)[3]
			E = 1 VAR1 (sweep) source function
			= 2 VAR2 (step) source function
			= 3 Constant (fixed) source function[3]
			= 4 VAR1' source function
	VS	VS channel definition[8]	VSA, 'BBBBBB', C
			A = n, for voltage source VSn[2, 12]
			BBBBBB = Voltage source name
			C = 1 VAR1 source function
			= 2 VAR2 source function
			= 3 Constant (fixed) source function
			= 4 VAR1' source function
	VM	VM channel definition[9]	VMA,'BBBBBB'
			A = n, for voltmeter VMn[2, 12]
			BBBBBB = Voltmeter name

Table 9-2 (continued)
System mode commands

Page[1]	Command	Function	Command String
SS	VR IR	VAR1 setup	AAB, ±CCC.CCCC, ±DDD.DDDD, ±EEE.EEE, ±FFF.FFFF
			AA = VR Voltage source (SMU, VS1...VS8)
			= IR Current source [SMU (only)]
			B = 1 Linear sweep
			= 2 Log 10 sweep
			= 3 Log 25 sweep
			= 4 Log 50 sweep
			±CCC.CCCC = -210.00 to +210.00 Start value (volts)[10]
			= -0.1050 to +0.1050 Start value (amps), 4200-SMU
			= -1.0500 to +1.0500 Start value (amps), 4210-SMU
			±DDD.DDDD = -210.00 to +210.00 Stop value (volts)
			= -0.1050 to +0.1050 Stop value (amps), 4200-SMU
			= -1.0500 to +1.0500 Stop value (amps), 4210-SMU
			±EEE.EEEE = -210.00 to +210.00 Step value (volts)[4, 10]
			= -0.1050 to +0.1050 Step value (amps), 4200-SMU
			= -1.0500 to +1.0500 Step value (amps), 4210-SMU
			±FFF.FFFF = -210.00 to +210.00 Compliance value (volts)[5]
			= -0.1050 to +0.1050 Comp value (amps), with 4200-SMU[5]
			= -1.0500 to +1.0500 Comp value (amps), with 4210-SMU[5]
	VP IP	VAR2 setup	AA ±BBB.BBBB, ±CCC.CCCC, DD, ±EEE.EEEE, FF
			AA = VP Voltage source (SMU, VS1...VSn)
			= IP Current source (SMU only)
			±BBB.BBBB =-210.00 to +210.00 Start value (volts)[10]
			= -0.1050 to +0.1050 Start value (amps), 4200-SMU
			= -1.0500 to +1.0500 Start value (amps), 4210-SMU
			±CCC.CCCC = -210.00 to +210.00 Step value (volts)[10]
			= -0.1050 to +0.1050 Step value (amps), 4200-SMU
			= -1.0500 to +1.0500 Step value (amps), 4210-SMU
			DD = 1 to 32 Number of steps in sweep
			±EEE.EEEE =-210.00 to +210.00 Compliance value (volts)[5]
			= -0.1050 to +0.1050 Comp value (amps), with 4200-SMU[5]
			= -1.0500 to +1.0500 Comp value (amps), with 4210-SMU[5]
			FF = 1 to 4 VAR2 source stepper index (optional)[13]
	ST	Auto Standby	ST A, B
			A = 1 to 8 (SMU channel number) B = 0 (disable auto standby) or 1 (enable auto standby)

Table 9-2 (continued)
System mode commands

Page[1]	Command	Function	Command String
SS (cont.)	VL IL	List sweep	AAB, C, ±DDD.DDDD, ±EE.EEE, ... ±EE.EEEE
			AA = VL Voltage source (SMU, VS1...VS8)
			= IL Current source [SMU (only)]
			B = 1 through 8Channel number
			C = 0 Slave
			= 1 Master
			±DDD.DDDD = -210.00 to +210.00 Compliance value (volts)[5]
			= -0.1050 to +0.1050 Comp value (amps), with 4200-SMU[5]
			= -1.0500 to +1.0500 Comp value (amps), with 4210-SMU[5]
			±EE.EEEE =Up to 4096 comma delimited sweep points (1, 2, 3, and so on)
	RT FS	VAR1' setup:	
		Set VAR1' Ratio	RT ±AA.A [,B](brackets indicate that B parameter is optional)
			±AA.A = -10 to +10 Ratio value
			B = 1 to 8 SMU channel number for ratio
		Set VAR1' Offset	FS ±AAA.A [,B](brackets indicate that B parameter is optional)
			±AAA.A = -210 to +210 Offset value
			B = 1 to 8 SMU channel number for offset
	VC IC	Constant SMUs setup	AAB, ±CCC.CCCC, ±DDD.DDDD
			AA = VC Voltage source
			= IC Current source
			B = 1 to 8 SMU channel number
			±CCC.CCCC = -210.00 to +210.00 Output value (volts)
			= -0.1050 to +0.1050 Output value (amps), 4200-SMU
			= -1.0500 to +1.0500 Output value (amps), 4210-SMU
			±DDD.DDDD = -210.00 to +210.00 Compliance value (volts)[5]
			= -0.1050 to +0.1050 Comp value (amps), with 4200-SMU[5]
			= -1.0500 to +1.0500 Comp value (amps), with 4210-SMU[5]
	SC	Constant VS setup	SCA, ±BBB.BBBB
			A = n, for voltage sourceVSn[2, 12]
			BBB.BBBB = -210.00 to +210.00 Output value (volts)
	HT	Set sweep hold time	HT AAA.A
			AAA.A = 0 to 655.3 Hold up start of sweep (sec)
	DT	Set sweep delay time	DT A.AAA
			A.AAA = 0 to 6.553 Source settle time (sec)
SM	WT	Set wait time	WT AAA.AAA

Table 9-2 (continued)
System mode commands

Page[1]	Command	Function	Command String
			AAA.AAA=0 to 100 Hold up start of test (sec)
	IN	Set interval	IN AA.AA
			AA.AA = 0.01 to 10 Time (in sec) between measurements (4145 emulation mode)
			= 0 to 10 Time (in sec) between measurements (4200 extended mode *only*)
	NR	Select number of readings	NR AAAA
			AAAA = 1 to 1024 Number of rdgs (4145 emulation mode)
			= 1 to 4096 Number of rdgs (4200 extended mode *only)*
	DM	Select display mode	DMA
			A = 1 Graphics display mode
			= 2 List display mode
	LI	List mode - Enables SMU channels (CH), V-sources (VS) and V-meters (VM) for a test sequence	LI ‘AAAAAA’, ‘AAAAAA’, ‘AAAAAA’, ...
			AAAAAA = A name assigned for CH, VS, or VM (see DE page) Note: Up to six names can be specified by the LI command
	XN	Configure graph X axis for electrical parameters	XN 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
			AAAAAA= Axis name (an SMU channel that is specified on DE page)
			B = 1 Linear scale
			= 2 Log scale
			±CCCC.CCC = ±9999 volts Minimum value
			= ±999 amps
			±DDDD.DDD = ±9999 volts Maximum value
			= ±999 amps
	XT	Configure graph X axis for time domain	XT AAAA.AA, BBBB.BB
			AAAA.AA= 0.01 to 9999 seconds Minimum value
			BBBB.BB = 0.01 to 9999 seconds Maximum value

Table 9-2 (continued)
System mode commands

Page[1]	Command	Function	Command String
SM	YA	Configure graph Y1 axis	YA 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
			AAAAAA= Axis name (an SMU channel that is specified on DE page)
			B = 1 Linear scale
			= 2 Log scale
			= 3 Log scale absolute value
			±CCCC.CCC = ±9999 volts Minimum value
			= ±999 amps
			±DDDD.DDD = ±9999 volts Maximum value
			=±999 amps
	YB	Configure graph Y2 axis	YB 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
			AAAAAA= Axis name (SMU channel that is specified on DE page)
			B = 1 Linear scale
			= 2 Log scale
			= 3 Log scale absolute value
			±CCCC.CCC = ±9999volts Minimum value
			= ±999 amps
			±DDDD.DDD = ±9999 volts Maximum value
			= ±999 amps
MD	ME	Control measurements	MEA
			A = 1 Trigger test, store readings in cleared buffer
			= 2 Trigger test, store readings in cleared buffer
			= 3 Trigger test, append readings to buffer
			= 4 Abort test
Any[6]	DO	Obtain output data	DO 'AAAAAA'
			AAAAAA = Name of measurement channel
			DO 'CHnT'where: n = Absolute channel number
	SR	Fixed source ranging	SR A,B
			A = The channel controlled: a value between 1 and the number of channels in the system (8 max).
			B = 0 Auto
			= 1 Best fixed
			> 0 to 1.0 Fixed range

Table 9-2 (continued)
System mode commands

Page[1]	Command	Function	Command String
Any	SV	Save file[7]	SV 'A BBBBBB CCCCCCCC'
			A = P Program file
			= D Data/Program file
			BBBBBB = Name of file (up to 6 characters)
			CCCCCCCC = Comment (up to 8 characters)
	GT	Get file[7]	GT 'A BBBBBB'
			A = P Program file
			= D Data/Program file
			BBBBBB = Name of saved file (up to 6 characters)
	MP	Map channel “n” to a given VS, SMU, or VM function	MP A, BBBC
			A = The channel to be mapped: a value between 1 and the number of channels in the system (8 max).
			BBB = SMU, VS, or VM[11]
			C = The number of the SMU, VS, or VM[11]

[1] The appropriate page must be selected before sending a command string. Commands to select a page:
DE = Channel definition page
SS = Source setup page
SM = Measurement setup page
MD = Measurement control page
US = User mode page

[2] If nothing is specified after the channel number, the channel is turned off.

[3] When the source mode is Common (3), the source function must be set to Constant (3).

[4] If performing a logarithmic sweep (B = 2, 3, or 4), do not set a step value.

[5] If sourcing voltage, you will be setting current compliance. If sourcing current, you will be setting voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum-allowable value: 100 pA with a PreAmp installed and 100nA without a PreAmp: KXCI sets it to the minimum allowable value.

[6] The command string can be sent in any system mode page.

[7] As shown, the parameters of the file command strings must be enclosed in single quotes. Each parameter must be separated by a space.

[8] To source voltage using the 4145B VS1...VSn function, define one of the Model 4200-SCS SMUs to emulate the VS.

[9] To measure voltage using the 4145B VM1...VMn function, define one of the Model 4200-SCS SMUs to emulate VM (note: if you do not define one of the Model 4200-SCS SMUs to emulate a VM, attempts to measure voltages through the nonexistent VM result in data values of 9.000E+37).

[10] If you specify a voltage start or step value below 0.001V, KXCI automatically sets the value to zero.

[11] If BBB and C values are not included in the command, the function defaults to SMU<A>, where <A> is the number of the channel to be mapped.

[12] The assigned “n” value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.

[13] If left blank, the default is 1 (first stepper). Use CH command to define 1 or more VAR2 sources. First defined VAR2 is index = 1. Second channel defined as VAR2 is index = 2, and so on up to a maximum of 4 VAR2 sources. Note that this is an extension to the traditional VAR2 capability.

Table 9-3
User mode commands

Page[1]	Command	Function	Command String
US	DV DI	SMU setup	AAB, CC, ±DDD.DDDD, ±EEE.EEEE
			AA = DV Voltage source
			= DI Current source
			B = 1, 2, ... or n SMU channel number[3] The largest permissible value of n equals the number of channels in the system (8 max).
			CC = 0 Autorange
			Fixed ranges, 4200-SMU/4210-SMU with PreAmp:
			CC = 1 1 nA or 20 V range
			= 2 10nA or 200 V range
			= 3 100nA or 200 V range
			= 4 1µA range or 200 mV range
			= 5 10µA range or 2V range
			= 6 100µA range
			= 7 1mA range
			= 8 10 mA range
			= 9 100 mA range
			= 10 1A (4210-SMU only)
			= 11 1pA
			= 12 10 pA
			= 13 100 pA
			Fixed ranges, 4200-SMU/4210-SMU without PreAmp:
			CC = 1 20 V
			= 2 200 V
			= 3 100nA or 200 V range
			= 4 1µA range
			= 5 10µA range
			= 6 100µA range
			= 7 1mA range
			= 8 10 mA range
			= 9 100 mA range
			= 10 1A (4210-SMU only)
			±DDD.DDDD=-210.00 to +210.00Output value (volts)
			= -0.1050 to +0.1050Output value (amps), 4200-SMU
			= -1.0500 to +1.0500Output value (amps), 4210-SMU
			±EEE.EEEE=-210.00 to +210.00Compliance value (volts)[2]
			= -0.1050 to +0.1050Compliance value (amps), 4200-SMU[2]
			= -1.0500 to +1.0500Compliance value (amps), 4210-SMU[2]
	DS	VS1...VS8 setup	DSA, ±BBB.BBBB
			A = n, for voltage sourceVSn[3]

Table 9-3 (continued)
User mode commands

Page[1]	Command	Function	Command String
			±BBB.BBBB =-20.00 to +20.00 Output value (volts)
US (cont.)	TV TI	Triggering	AABB
			AA = TV Voltage measurement
			= TI Current measurement
			TV *TI*
			BB = 1 SMU1 SMU1
			= 2 SMU2 SMU2
			= 3 SMU3 SMU3
			= 4 SMU4 SMU4
			= 5 VM1 SMU5
			= 6 VM2 SMU6
			= 7 SMU5 SMU7
			= 8 SMU6 SMU8
			= 9 SMU7
			= 10 SMU8
			= 11 VM3
			= 12 VM4
			= 13 VM5
			= 14 VM6
			= 15 VM7
			= 16 VM8

[1] In order to send user mode commands, the user mode page must be selected. This page is selected by sending the US command.
[2] If sourcing voltage, you will be setting current compliance. If sourcing current, you will be setting voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum-allowable value: 100 pA with a PreAmp installed and 100nA without a PreAmp, KXCI sets it to the minimum allowable value.
[3] The assigned "n" value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.

Table 9-4
Commands common to system and user modes

Command	Function	Command String
IT	Set integration time	ITA
		A = 1 Short
		= 2 Medium
		= 3 Long
		= 4,X,Y,Z Custom (only available in 4200 extended mode):
		X = 0.0 to 100 Delay factor[1]
		Y = 0.0 to 100 Filter factor[1]
		Z = 0.01 to 100 A/D converter integration time, in number of power-line cycles (PLCs)[1]

Table 9-4 (continued)
Commands common to system and user modes

Command	Function	Command String
ID	Places the ID of the instrument: for the 4200 extended mode or the 4145 emulation mode: in a buffer.	ID
DR	Control service request for "Data Ready."	DRA
		A = 0 Disable service request for data ready
		= 1 Enable service request for data ready
BC	Clear data buffer, and bit B0 (Data Ready) of status byte.	BC
RS	Set the measurement resolution for all channels.	RS A
		A = Resolution, in number of digits: 3 to 7 digits in 4200 extended mode 3 to 5 digits in 4145 emulation mode
RI	Instructs a SMU to go the the specified range immediately.	RI A, B, C
		A = 1, 2, ... or n SMU channel number. The largest permissible value of n equals the number of channels in the system (8 max).
		B = Range, permitted values: 100 E-9 to 1 amps, 4210-SMU without a PreAmp 1 E-12 to 1 amps, 4210-SMU with a PreAmp
		C = Compliance, permitted values: 10% to 100% of range.
RG	Set the lowest current range to be used when measuring.	RG A,B
		A = 1 to 8SMU channel number.
		B = The lowest auto-ranged range2. Permitted values: - 100 E-9 to 100 E-3 amps, 4200-SMU without a PreAmp - 1 E-12 to 100 E-3 amps, 4200-SMU with a PreAmp - 100 E-9 to 1 amps, 4210-SMU without a PreAmp - 1 E-12 to 1 amps, 4210-SMU with a PreAmp
AC	Perform auto calibration.	AC A
		A = 1 to 8 SMU channel number.
EC	Set exit on compliance.	EC A
		A = 0 Will not exit (OFF)
		= 1 Will exit (ON)

Table 9-4 (continued)
Commands common to system and user modes

Command	Function	Command String
EM	Switch between 4145 and 4200 modes.	EM A,B
		A = 0 4145 mode
		= 1 4200 mode
		B = 0 This session Only
		= 1 This session and ALL subsequent sessions (writes to KCON file)

[1]For information about these factors, refer to Section 6.
[2]The default auto-ranged ranges are as follows: 100 nA without a preamp and 1 nA with a preamp.

Table 9-5
4200 extended mode-only commands

Command	Function	Command String
*OPT?	Get the KXCI configuration of the Model 4200.	*OPT?

GPIB command reference

GPIB commands to control instrument operation are divided into three categories:

- **System mode commands**: This comprehensive set of commands allows you to utilize all the source-measure capabilities of the SMUs installed in Model 4200-SCS.
- **User mode commands**: This limited set of commands allows you to perform basic source-measure operation.
- **Common commands:** These commands are valid in any operating mode.
- **4200 extended mode-only commands**: These commands are valid only while using the 4200 extended mode (vs. the 4145 emulation mode).

NOTE *Numeric values can be entered in fixed decimal format (for example, 0.1234) or floating decimal format (for example, 123.4E-3). Maximum number of characters for the value is 12. The maximum number of digits for an exponent is two.*

System mode commands

Most system mode commands are divided into groups, known as pages. In order to use these commands, the appropriate page has to be selected. System mode commands are grouped as follows. The command to select each page is shown in parentheses.

- Channel definition page (DE)
- Source setup page (SS)
- Measurement setup page (SM)
- Measurement control page (MD)
- Data output and file commands (valid in any system mode page)

Channel definition page (DE)

The command strings for the DE page are used for the following operations:

- SMU channel definition

- VS1...VSn channel definition
- VM1...VMn channel definition

In order to send the following command strings to the Model 4200-SCS, the channel definition page must first be selected by sending the following command:

```
DE
```

CH Command: SMU channel definition

For every used channel that is configured as an SMU (see KXCI settings in KCON), you have to specify names for voltage and current, select the source mode (voltage, current, or common), and select the source function (VAR1, VAR2, constant, or VAR1').

A channel can be disabled by sending the following command:

```
CHx
```

Where x = 1-8, the SMU channel to be disabled. For example, to disable channels 1 through 4, send the following command string:

```
CH1; CH2; CH3; CH4
```

The VAR1 source function is used to perform a linear or logarithmic sweep. The VAR1 function performs a sweep that is synchronized to the steps of VAR2. The VAR1 sweep is performed whenever VAR2 goes to a new step value. The constant source function outputs a fixed (constant) source value.

The VAR1' source function is similar to the VAR1 function, except that each sweep step is scaled by the Ratio value (RT) and an Offset (FS) as follows:

VAR1' sweep step = (VAR1 sweep step × RT) + FS

For example, assume VAR1 is set to sweep from +1V to +3V using 1V steps. If Ratio (RT) is set to 2, and Offset (FS) is set to 1, each step of VAR1' is calculated as follows:

VAR1' step 1 = (1V × 2) + 1 = 3V
VAR1' step 2 = (2V × 2) + 1 = 5 V
VAR1' step 3 = (3V × 2) + 1 = 7V

Use the following command string to define the channel of each SMU:

```
CHA, 'BBBBBB', 'CCCCCC', D, E
```

CH = Command string prefix

SMU channel[1]

A = 1, 2, ... or n SMU channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Voltage name

BBBBBB = User specified name (up to six characters enclosed in single quotes)

Current name

CCCCCC = User specified name (up to six characters enclosed in single quotes)

Source mode[2]

D = 1 Voltage source
= 2 Current source
= 3 Common (output high connected to common)

Source function[2]

1. If nothing is specified after the prefix and channel number (that is, CH2 term), the channel is turned off (not used).
2. When the source mode is set to 3 (common), source function must be set to 3 (constant).

E = 1 VAR1
= 2 VAR2
= 3 Constant
= 4 VAR1'

Example

The following command string sets up the SMU assigned to channel 3 to source a fixed voltage (1V). The specified names for voltage and current are V1 and I1 respectively.

```
CH3, 'V1', 'I1', 1, 3
```

VS Command: VS1...VSn channel definition

KXCI allows up to eight source-measure units to function solely as voltage sources. Any channel may be used for any voltage-source function between VS1 and VS8. For example, in a system containing four SMUs, you can use SMU2 as VS5. For details on KXCI settings, see Section 7.

For each voltage source that is used, you have to specify a name and select the source function (VAR1, VAR2, constant, or VAR1'). The VAR1 function performs a voltage sweep that is synchronized to the steps of VAR2. The VAR1 source function is used to perform a linear or logarithmic voltage sweep. The VAR1 sweep is performed whenever VAR2 goes to a new step value. The VAR1' function is the same as VAR1 except that each step of the sweep is scaled by a specified Ratio (RT) and Offset (FS).

The constant source function outputs a fixed (constant) voltage source value. More information on source functions is available throughout this section.

Use the following command string to define the channel(s) used by the voltage source(s):

```
VSA, 'BBBBBB', C
```

VS = Command string prefix

VS channel[3]

A = n, for voltage source VSn.[4]

Voltage source name

BBBBBB = User specified name (up to 6 characters enclosed in single quotes)

Voltage Source function

C = 1 VAR1
= 2 VAR2
= 3 Constant
= 4 VAR1'

Example

The following command string sets up the channel used by VS1 to perform a voltage sweep:

```
VS1, 'VS1', 1
```

3. If nothing is specified after the prefix and channel number (that is, VS2 term), the channel is turned off (not used).
4. The assigned “n” value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.

VM Command: VM1...VMn channel definition

KXCI allows up to eight source-measure units (SMUs) to function solely as voltmeters. Any channel may be used for any voltmeter function between VM1 and VM8.[5] For example, in a system containing four SMUs, you can use SMU3 as VM7. See KXCI configuration for details.

Use the following command string to define the channel(s) used for the voltmeter(s):

```
VMA, 'BBBBBB'
```

VM = Command string prefix

Voltmeter channel[6]

A = n, for voltmeter channel VMn.[7]

Voltmeter name

BBBBBB = User specified name (up to 6 characters enclosed in single quotes)

Example

The following command string defines the channel used for VM1 (specifies the name VM1):

```
VM1, 'VM1'
```

Source setup page (SS)

The command strings for the SS page are used for the following operations:

- VAR1 setup
- VAR2 setup
- VAR1' setup
- List sweep setup
- Constant SMUs setup
- Constant VS setup
- Set sweep hold time
- Set sweep delay time

In order to send the following command strings to the Model 4200-SCS, the source setup page must first be selected by sending the following command:

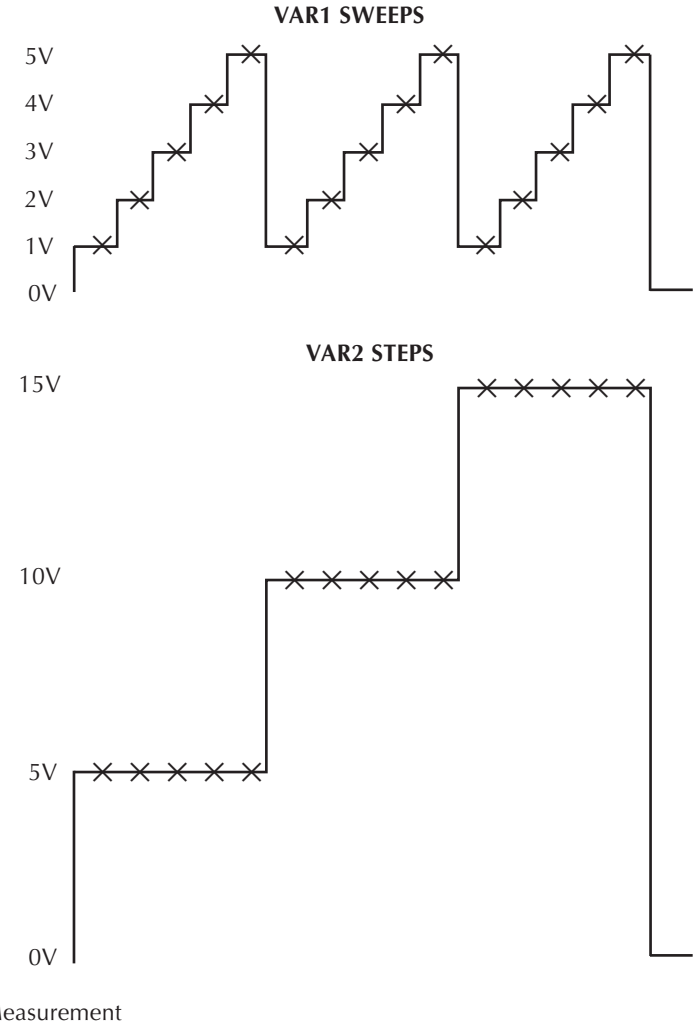
```
SS
```

5. If you do not define one of the Model 4200-SCS SMUs to emulate a VM, attempts to measure voltages through the nonexistent VM result in data values of 9.000E+37.
6. If nothing is specified after the prefix and channel number (that is, VM2 term), the channel is turned off (not used).
7. The assigned “n” value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.

VR and IR commands: VAR1 setup

When VAR1 is a selected source function, it will perform a sweep that is synchronized to the steps of the VAR2 step function. The VAR1 sweep is repeated whenever VAR2 goes to a new step value. See Figure 9-10.

Figure 9-10
Illustration of synchronized VAR1 Sweeps and VAR2 steps



× = Measurement

If the source that is being used is an SMU, the source mode for the sweep can be voltage or current. If, however, a voltage source (VS1...Sn) is being used, the source mode must be voltage.

The sweep can be performed on a linear or logarithmic scale. With the linear sweep mode selected, the start, stop, and step value parameters define the sweep. Each VAR1 sweep in Figure 9-10 sweeps from 1V (start) to 5 V (stop) in 1V steps.

With a logarithmic sweep mode selected (log base 10, 25 or 50), only the start and stop values must be specified. Step size is automatically set to provide a symmetrical sweep on the logarithmic scale.

NOTE *The time spent on each sweep step depends on the user-set delay time and the time it takes to perform the measurement.*

The start of the sweep can be delayed by setting a hold time.

Use the following command string to configure the VAR1 sweep:

```
AAB, ±CCC.CCCC, ±DDD.DDDD, ±EEE.EEEE, ±FFF.FFFF
```

Source mode

AA = VRVoltage source (SMU or VS1...VS8)
= IR Current source (SMU only)

Sweep mode

B = 1 Linear sweep
= 2 Log10 sweep
= 3 Log25 sweep
= 4 Log50 sweep

Start value[8]

±CCC.CCCC = -210.00 to +210.00 (volts)[12]
= -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU

Stop value[8]

±DDD.DDDD = -210.00 to +210.00 (volts)
= -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU

Step value [8, 9, 10]

±EEE.EEEE = -210.00 to +210.00 (volts)[11]
= -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU

Compliance value [12]

±FFF.FFFF = -210.00 to +210.00 (volts)
= -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU

Example

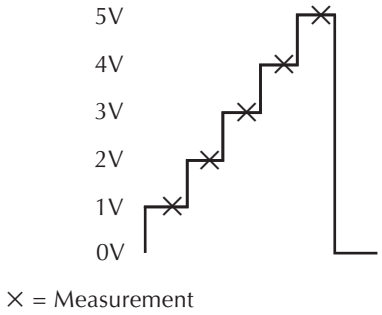
The following command string sets up a VAR1 linear sweep (start = 1V, stop = 5 V, step = 1V, and compliance = 10 mA):

```
VR1, 1, 5, 1, 0.01
```

Figure 9-11 shows the sweep that results from this setup. Figure 9-10 shows the results of the same setup when used with a VAR2 step command.

8. With the voltage source mode (VR) specified in the command string, the output value will be in volts. For the current source mode (IR), the output value will be in amps.
9. If the sweep mode (B) is set for a log sweep, do not set a step value (±EEE.EEEE).
10. Max number of points for VAR1 is 1024. Number of points = (int)(Abs((Stop Value - Start Value) / Step Value) + 1.5).
11. I f you specify a voltage start or step value below 0.001V, KXCI automatically sets the value to zero.
12. With the voltage source mode (VR) specified in the command string, you will be setting the current compliance. For the current source mode (IR), you will be setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum allowable value (100 pA with a PreAmp installed and 100nA without a PreAmp), KXCI sets it to the minimum allowable value.

Figure 9-11
Sweep resulting from the VR1, 1, 5, 1, 0.01 command string

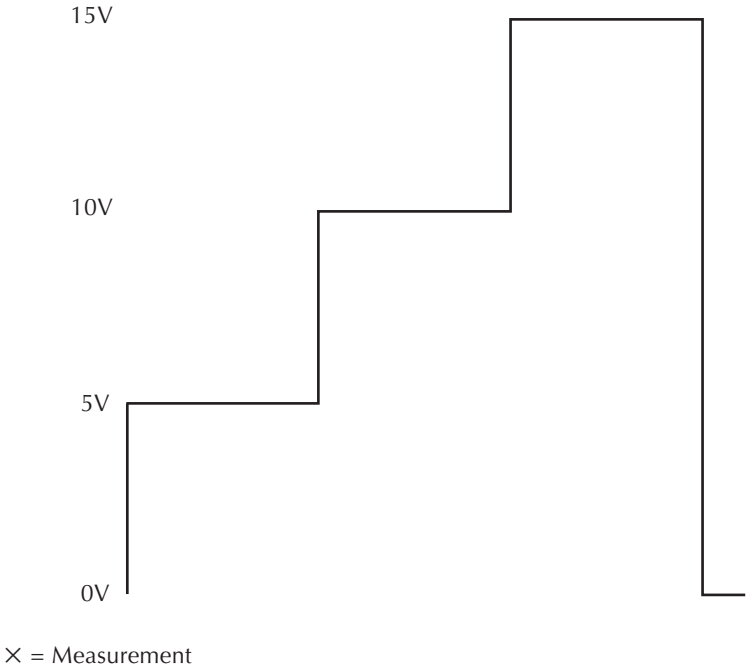


VP and IP commands: VAR2 setup

If the source that is being used is an SMU, the source mode for the VAR2 steps can be voltage or current. If, however, a voltage source (VS1...VS8) is being used, the source mode must be voltage (for configuration details, refer to Section 7).

Parameters for the VAR2 step function include the start value, step value, and the number of steps. In Figure 9-12 the VAR2 step function starts at 5 V, steps in 5 V increments and has 3 steps.

Figure 9-12
Steps resulting from the VP 5, 5, 3, 0.01 command string



Use the following command string to configure the VAR2 sweep:

```
AA ±BBB.BBBB, ±CCC.CCCC, DD, ±EEE.EEEE
```

Source mode

AA	= VP	Voltage source (SMU or VS1...VS8)
	= IP	Current source (SMU only)

Start value[13]

±BBB.BBBB	=	-210.00 to +210.00 (volts)[14]
	=	-0.1050 to +0.1050 (amps), 4200-SMU
	=	-1.0500 to +1.0500 (amps), 4210-SMU

Step value[13]

±CCC.CCCC	=	-210.00 to +210.00 (volts)[14]
	=	-0.1050 to +0.1050 (amps), 4200-SMU
	=	-1.0500 to +1.0500 (amps), 4210-SMU

Number of steps

DD	=	1 to 32

Compliance value[15]

±EEE.EEEE	=	-210.00 to +210.00 (volts)
	=	-0.1050 to +0.1050 (amps), 4200-SMU
	=	-1.0500 to +1.0500 (amps), 4210-SMU

Example

The following command string sets up a VAR2 voltage sweep with start = 5 V, step = 5 V, number of steps = 3, and compliance = 10 mA:

```
VP 5, 5, 3, 0.01
```

This command string performs the steps shown in Figure 9-12. These are the same steps that are shown synchronized with sweeps, in Figure 9-10.

ST command: Auto standby

For an SMU, the ST command is used to control auto standby. When auto standby is enabled (1), the SMU automatically goes into standby when the test completes. When disabled (0), the output stays on when the test completes.

Use the following command string to set auto standby for an SMU:

```
ST A, B
```

SMU channel

A = 1 to 8	SMU channel number

SMU output

B = 0	Disable auto standby
= 1	Enable auto standby

13. With the voltage source mode (VP) specified in the command string, the output value will be in volts. For the current source mode (IP), the output value will be in amps.
14. If you specify a voltage start or step value below 0.001V, the value is automatically set to zero.
15. With the voltage source mode (VP) specified in the command string, you will be setting the current compliance. For the current source mode (IP), you will be setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum allowable value (100 pA with a PreAmp installed and 100nA without a PreAmp), KXCI sets it to the minimum allowable value.

RT and FS commands: VAR1' setup

When VAR1' is a selected source function, it will perform the VAR1 sweep with each step scaled by the Ratio (RT) value and Offset (FS) value as follows:

VAR1' sweep step = (VAR1 sweep step × RT) + FS

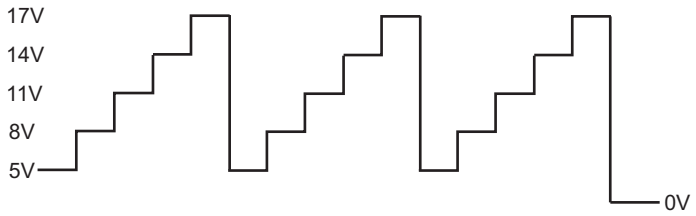
Unique RT and FS values can be assigned to any available SMU channel in the system.

The Figure 9-11 VAR1 sweep has five steps: 1V, 2V, 3V, 4V, and 5 V. The corresponding VAR' sweep has the following steps when RT=3 and FS = 2:

VAR1' step 1 = (1V × 3) + 2 = 5 V
VAR1' step 2 = (2V × 3) + 2 = 8V
VAR1' step 3 = (3V × 3) + 2 = 11V
VAR1' step 4 = (4V × 3) + 2 = 14V
VAR1' step 5 = (5 V × 3) + 2 = 17V

This VAR1' sweep (not drawn to scale) is illustrated in Figure 9-13.

Figure 9-13
VAR' sweep when RT=3, FS=2, and the VAR1 sweep is as shown in Figure 9-11



Use the following command strings to configure the VAR1' sweep:

Ratio

```
RT ±AA.A[,B]
```

```
AA.A = -10 to 10
```

B = 1 to 8 Available SMU channel

Offset

```
FS ±AAA.A[,B]
```

AAA.A = -210 to 210

B = 1 to 8 Available SMU channel

NOTE *The brackets that enclose the B parameter ([,B]) indicate that it is optional. If using the B parameter, do not include the brackets in the command string (see following Example).*

With the B parameter omitted, ratio and offset will apply to all channels configured to VAR1'.

Example

The following command strings set up the VAR1' sweep that was illustrated in Figure 9-13 (ratio = 3, offset = 2):

```
RT +3,2 Ratio
FS +2,2 Offset
```

The above commands set up VAR1' for SMU Channel 2. When Channel 2 is defined for a VAR1' sweep, RT will set to 3 and FS will set to 2.

VL and IL commands: List sweep setup

When list sweep is a selected source function, it will perform a list sweep with arbitrary sweep points in order of the defined sweep parameters. If the source that is being used is an SMU, the source mode for the sweep can be voltage or current. If, however, a voltage source (VS1...Sn) is being used, the source mode must be voltage.

Use the following command string to configure the List sweep:

```
AAB, C, ±DDD.DDDD, ±EE.EEEE, ... ±EE.EEEE
```

Source mode

AA	= VL Voltage source (SMU or VS1...VS8) = IL Current source (SMU only)

Channel number

B	= 1 through 8

Master/slave mode

C	= 0 Slave mode = 1 Master mode

Compliance value[16]

±DDD.DDDD	= -210.00 to +210.00 (volts) = -0.1050 to +0.1050 (amps), 4200-SMU = -1.0500 to +1.0500 (amps), 4210-SMU

List values[17]

±EE.EEEE	= -210.00 to +210.00 (volts) = -0.1050 to +0.1050 (amps), 4200-SMU = -1.0500 to +1.0500 (amps), 4210-SMU

Example

The following command string sets up a channel 1 List sweep (1V, 5 V, 2V arbitrary steps, and compliance = 10 mA):

```
VL1,1, 0.01, 1, 5, 2
```

16. With the voltage source mode (VR) specified in the command string, you will be setting the current compliance. For the current source mode (IR), you will be setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum allowable value, 100 pA with a PreAmp installed and 100nA without a PreAmp, KXCI sets it to the minimum allowable value.
17. With the voltage source mode (VL) specified in the command string, the output value will be in volts. For the current source mode (IL), the output value will be in amps. Maximum number of points for List is 4096. Sweep points must be delimited by commas.

VC and IC commands: SMU constant voltage or current setup

For any channel configured as an SMU (see KCON configuration), use the following command string to configure the SMU to output a fixed (constant) voltage or current level:

```
AAB, ±CCC.CCCC, ±DDD.DDDD
```

Source mode

AA = VC Voltage source
= IC Current source

SMU channel

B = 1 to 8 SMU channel number

Output value[18]

±CCC.CCCC = -210.00 to +210.00 (volts)
= -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU

Compliance value[19]

±DDD.DDDD = -0.1050 to +0.1050 (amps), 4200-SMU
= -1.0500 to +1.0500 (amps), 4210-SMU
= -210.00 to 210.00 (volts)

SC command: Constant VS setup

For any channel configured solely as a VS1...VSn voltage source (see KCON configuration), use the following command string to configure the source to output a fixed (constant) voltage level:

```
SCA, ±BBB.BBBB
```

SC Command string prefix

VS channel

A = n, for voltage source VSn[20]

Output voltage value

±BBB.BBBB = -210.00 to +210.00

Example

The following command string sets up VS1 to output a constant 20 V level:

```
SC1, 20
```

18. With the voltage source mode (VC) specified in the command string, the output value will be in volts. For the current source mode (IC), the output value will be in amps.
19. With the voltage source mode (VC) specified in the command string, you set the current compliance. For the current source mode (IC), you set the voltage compliance.
20. The assigned “n” value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.

HT command: Set sweep hold time

The start of a sweep can be delayed by setting a hold time. When the sweep is triggered, it will start after the hold time period expires. Use the following command string to set the hold time:

```
HT AAA.A
```

HT = Command string prefix

Hold time

AAA.A = 0 to 655.3 Hold time in seconds

Example

The following command string sets hold time to one second:

```
HT 1
```

DT command: Set sweep delay time

For a VAR1 sweep, the time duration spent on each step of the sweep is determined by the user set delay time and the time it takes to perform the measurement.

The delay time is typically used to allow the source to settle before performing the measurement. For example, assume a delay time of 1sec. At each step of the sweep, the source will be allowed to settle for 1sec before the measurement is taken.

Use the following command string to set delay time:

```
DT A.AAA
```

DT = Command string prefix

Delay time

A.AAA = 0 to 6.553 Delay time in seconds

Example

The following command string sets delay time to 1 sec:

```
DT 1
```

Measurement setup page (SM)

The command strings for the SM page are used for the following operations:

- Set wait time
- Set interval
- Select number of readings
- Select list display mode

In order to send the following command strings to the Model 4200-SCS, the SM page must first be selected by sending the following command:

```
SM
```

WT command: Set wait time

For time domain measurements, the start of a test sequence can be delayed by setting a wait time. The test sequence will start after the wait time period expires. Use the following command string to set the wait time:

```
WT AAA.AAA
```

WT = Command string prefix

Wait time

AAA.AAA = 0 to 100 Wait time in seconds

Example

The following command string sets wait time to 100 msec:

```
WT 0.1
```

IN command: Set interval

For time domain measurements, the time interval between sample measurements can be set by the user. After a sample measurement is performed, the next measurement will start after the time interval expires. Use the following command string to set interval:

```
IN AA.AA
```

IN = Command string prefix

Wait time

AA.AA = 0.01 to 10 Interval in seconds

Example

The following command string sets the interval to 100 msec:

```
IN 0.1
```

NR command: Select number of readings

For time domain measurements, up to 1024 sample measurements can be performed. The readings are stored in the buffer. Use the following command string to select the number of readings:

```
NR AAAA
```

NR = Command string prefix

Number of readings

AAAA = Number of measurements to perform:
1 to 4096 in 4200 extended mode
1 to 1024 in 4145 emulation mode

Example

The following command string sets up the Model 4200-SCS to perform 200 sample measurements:

```
NR 200
```

DM command: Select display mode

The Model 4200-SCS supports the HP 4145B graphics display mode and accepts the HP 4145B list display-mode command (the Model 4200-SCS does not accept the matrix mode and schmoo mode commands (DM3 and DM4)).

```
DMA
```

DM = Command string prefix

Display modes

A = 1 Graphics display mode
= 2 List display mode

Example

The following command string prepares the Model 4200-SCS to receive graphics commands:

```
DM1
```

LI command: List mode, enables V and I functions for test sequence

With the Model 4200-SCS in the list display mode (DM2 asserted), the LI command enables functions to be measured in a test sequence. A function is enabled by including the SMU channel name (as assigned by the CH command), V-source name (as assigned by the VS command), or the V-meter name (as assigned by the VM command) in the command string. The DE page is used to assign names to V and I functions.

Up to six names can be specified in the LI command string. Functions not specified (enabled) are not measured. Data sheet columns for disabled functions are not shown.

```
LI 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA', 'AAAAAA'
```

LI = Command string prefix

Name parameter

AAAAAA = A name assigned for CH, VS, or VM (see DE page)

Example

Assume the following names have been assigned using the DE page: For SMU channel 1 (CH1), voltage is named V1, current is named I1, and a voltage source (VS1) is named VS1.

The following command string enables the above functions for a test sequence:

```
L1 'V1', 'I1', 'VS1'
```

XN command: Configure graph X axis for electrical parameter

The following command string configures the X axis of the graph to plot an electrical parameter:

```
XN 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

XN = Command string prefix

X axis SMU channel name

AAAAAA = The SMU channel name for the X axis, up to 6 characters long. Must be one of the SMU channel names that you specify on the channel definition (DE) page.

X axis scale type

B = 1 Linear scale
= 2 Log scale

X axis minimum value

±CCCC.CCC = ±9999 (volts)
= ±999 (amps)

X axis maximum value

±DDDD.DDD = ±9999 (volts)
= ±999 (amps)

Example

The command string:

```
XN 'V1', 1, -5, 5
```

does the following:

- Specifies that values from SMU channel V1 are to be plotted on the X axis.
- Sets up the X axis to be scaled linearly between -5 V and +5 V.

XT command: Configure graph X axis for time domain

The following command string configures the X axis of the graph to plot time domain values (in seconds):

```
XT AAAA.AA, BBBB.BB
```

XT = Command string prefix

X-axis minimum time value

AAAA.AA = 0.01 to 9999 (seconds)

X-axis maximum time value

BBBB.BB = 0.01 to 9999 (seconds)

Example

The command string:

```
XT 0, 10
```

does the following:

- Specifies that time domain values are to be plotted on the X axis.
- Sets up the X axis to be scaled between 0 and 10 sec.

YA command: Configure graph Y1 axis

The following command string configures the Y1 axis of the graph:

```
YA 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

YA = Command string prefix

Y1 axis SMU channel name

AAAAAA = The SMU channel name for the Y1 axis, up to 6 characters long. Must be one of the SMU channel names that you specify on the channel definition (DE) page.

Y1 axis scale type

B = 1 Linear scale
= 2 Log scale
= 3 Log scale absolute value

Y1 axis minimum value

±CCCC.CCC = ±9999 (volts)
= ±999 (amps)

Y1 axis maximum value

±DDDD.DDD = ±9999 (volts
= ±999 (amps)

Example

The command string:

```
YA 'I1', 1, -5E-9, 5E-9
```

does the following:

- Specifies that values from SMU channel I1 are to be plotted on the Y1 axis.
- Sets up the Y1 axis to be scaled linearly between -5 nA and +5 nA.

YB command: Configure graph Y2 axis

The following command string (optional) configures the Y2 axis of the graph:

```
YB 'AAAAAA', B, ±CCCC.CCC, ±DDDD.DDD
```

YB = Command string prefix

Y2 axis SMU channel name

AAAAAA = The SMU channel name for the Y2 axis, up to 6 characters long. Must be one of the SMU channel names that you specify on the channel definition (DE) page.

Y2 axis scale type

B = 1 Linear scale
= 2 Log scale
= 3 Log scale absolute value

Y2 axis minimum value

±CCCC.CCC = ±9999 (volts)
= ±999 (amps)

Y2 axis maximum value

±DDDD.DDD = ±9999 (volts)
= ±999 (amps)

Example

The command string:

```
YB 'I2', 2, 100E-9, 1E-3
```

does the following:

- Specifies that values from SMU channel I2 are to be plotted on the Y2 axis.
- Sets up the Y2 axis to be scaled logarithmically between 100nA and 1mA.

Measurement control page (MD)

In order to send the command string to control measurements, the measurement control page must first be selected by sending the following command:

```
MD
```

ME command: Control measurements

There are four command strings to control measurements. The ME1 or ME2 command will trigger the start of the test and perform the programmed number of measurements. The measured readings are stored in the buffer. Note that the buffer is cleared before readings are stored.

The ME3 command also triggers the test but does not clear the buffer before storing the measured readings. The readings are appended to the readings already stored in the buffer. The buffer can hold up to 4096 readings.

The ME4 command aborts the test.

```
MEA
```

ME = Command string prefix

Control measurements

A = 1 SingleTrigger test, store readings in cleared buffer
= 2 RepeatTrigger test, store readings in cleared buffer
= 3 AppendTrigger test, append readings to buffer
= 4 StopAbort test

Example

The following command string triggers the start of the test and stores the readings in the cleared buffer:

```
ME1
```

Data output and file commands

The command strings for the following operations are valid in any system mode page:

- Obtain output data
- Save file
- Get file
- Map channel n to a given VS, SMU, or VM function

DO command: Obtain output data

After measurements are performed, the following command string is used to request the readings. After the Model 4200-SCS is addressed to talk, the readings are sent to the computer.

```
DO 'AAAAAA'
```

DO = Command string prefix

Measurement channel name

AAAAAA = User-specified name of the channel that performed the measurement, up to 6 characters long.

NOTE *To access the timestamp data that was acquired along with V or I measurements or both, use the following:*

```
DO 'CHnT'
```

where: n = Absolute channel number. Unlike V-and I-name strings, this label cannot be changed through the CH or VS commands.

Output data

After sending the DO command to request measurement data, and addressing the Model 4200 to talk, output data will be sent to the computer in the following format:

```
X±N.NNNN E±NN,  X±N.NNNN E±NN,  ... X±N.NNNN E±NN
```

X is the status of the data (where X = N for a normal reading), following by the reading mantissa and exponent. The delimiter can be set to CR/LF or Comma, and EOI can be enabled or disabled. For details on KXCI settings, see Section 7.

Example

Assuming one or more measurements were performed, the following command string will request the reading string for an SMU channel that is named Volt:

```
DO 'Volt'
```

SV command: Save file

The save command string is used to save a program file or data file. When saving a program file, the present instrument settings are stored in a file at the following directory path:

```
C:\S4200\sys\KXCI
```

The user specifies the name for the file.

NOTE *The get command string is used to acquire the saved file.*

Use the following command sequence to save a file:

```
SV 'A BBBBBB CCCCCCCC'
```

SV = Command string prefix

File type[21]

A = P Program file
= D Data/Program file

File name[21]

BBBBBB = Name of file (up to 6 characters)

21. As shown in the command string, file type, file name and the comment must be separated from each other by a space. Also note that these components of the command string must be enclosed in single quotes.

Comment[22]

CCCCCCCC = Comment (up to 8 characters)

Example

The following command string saves the command sequence as a program file named **Setup1**.

```
SV 'P Setup1'
```

GT command: Get file

The get command string is used to acquire (load) the saved data file or program file. For a program file it launches the program, and for a data file it merely opens the files. When the saved program file is recalled, the instrument returns to the settings stored in that file.

NOTE *The save command string is used to save instrument settings, or store data acquired in a test.*

Use the following command sequence to recall a file:

```
GT 'A BBBBBB'
```

GT = Command string prefix

File type[23]

A = P Program file
D Data file/program

File name[23]

BBBBBB = Name of file (up to 6 characters)

Example

The following command string gets the program file named **Setup1**:

```
GT 'P Setup1'
```

Channel mapping command

The following command may be used with any system mode page.

MP command: Map channel

The following command string maps channel n to a given VS, SMU, or VM function:

MP A, BBBC

A = The channel to be mapped: a value between 1 and the number of channels in the system (8 max).

BBB = SMU, VS, or VM[24]

C = The number of the SMU, VS, or VM[24]

22. The comment is optional. If not used, the space after the file name is not needed.
23. As shown in the command string, file type and file name must be separated by a space. Also note that these components of the command string must be enclosed in single quotes.
24. If BBB and C values are not included in the command, the function defaults to SMU<A>, where <A> is the number of the channel to be mapped.

Example

The following command string maps channel 3 to VM5:

```
MP 3, VM5
```

Fixed source ranging command

The following command may be used with any system mode page.

SR command: Set fixed source range

The following command sets a fixed source range on channel n:

```
SR A, B
```

A = The channel to be controlled: A value between 1 and the number of channels in the system (8 max).

B
= 0 Auto
= 2 Best fixed range (determined by maximum sweep parameters)
> 0 to 1.0Fixed range

The default setting is auto range for backwards compatibility. If you specify a range that is below the bias or sweep parameters that follow, the range is adjusted to accommodate the sweep.

Example

The following command string selects best fixed range on channel 1:

```
SR 1, 2
```

User mode commands (US)

The user mode (US) command strings are used for the following operations:

- SMU setup
- VS1...VS8 setup
- Triggering

In order to send these command strings to the Model 4200-SCS, the user mode must first be selected by sending the following command:

```
US
```

DV and DI commands: SMU setup

For every channel that is configured as an SMU, you will have to select the source mode (voltage or current) and source output range, and set the output and compliance values. Use the following command string to set up each SMU:

```
AAB, CC, ±DDD.DDDD, ±EEE.EEEE
```

Source mode

AA	= DV	Voltage source
	= DI	Current source

SMU channel

B	= 1 to 8	SMU channel number

Source range

Voltage source mode specified (AA = DV):

CC	= 0	Autorange
	= 1	20 V range
	= 2	200 V range
	= 3	200 V range

Current source mode specified (AA = DI):

CC	= 0	Autorange	CC	= 5	10µA range	CC = 10	1A range**
	= 1	1nA range*		= 6	100µA range	= 11	1pA range*
	= 2	10nA range*		= 7	1mA range	= 11	10 pA range*
	= 3	100nA range		= 8	10 mA range	= 12	100 pA range*
	= 4	1µA range		= 9	100 mA range		

* Only with a preamp

** Only with a 4210-SMU

Output value[25]

±DDD.DDDD	=	-210.00 to +210.00 (volts)
	=	-0.1050 to +0.1050 (amps), 4200-SMU
	=	-1.0500 to +1.0500 (amps), 4210-SMU

Compliance value[26]

±EEE.EEEE	=	-210.00 to 210.00 (volts)
	=	-0.1050 to +0.1050 (amps), 4200-SMU
	=	-1.0500 to +1.0500 (amps), 4210-SMU

Example

The following command string configures SMU1 to source 10 V on the 20 V source range, and sets current compliance to 10 mA:

```
DV1, 1, 10, 10E-3
```

25. With the voltage source mode (DV) specified in the command string, the output value will be in volts. For the current source mode (DI), the output value will be in amps.
26. With the voltage source mode (DV) specified in the command string, you will be setting the current compliance. For the current source mode (DI), you will be setting the voltage compliance. When sourcing voltage, note that if you specify a compliance current that is below the minimum-allowable value (100 pA with a PreAmp installed and 100nA without a PreAmp), KXCI sets it to the minimum allowable value.

DS Command: VS1...VS8 setup

KXCI allows up to eight source-measure units to function solely as voltage sources. Any channel may be used for any voltage-source function between VS1 and VS8. For example, in a system containing four SMUs, you can use SMU2 as VS5. For details on KXCI settings, see Section 7.

For each voltage source that is used, you have to specify the channel number and the voltage output value.

Use the following command string to set up each voltage source:

```
DSA, ±BBB.BBBB
```

DS = Command string prefix

Voltage source channel

A = n, for voltage source VSn[27]

Output voltage value

±BBB.BBBB = -210.00 to +210.00

Example

The following command string sets VS1 to output 20 V:

```
DS1, 20
```

TV and TI commands: Triggering

The trigger command string is used to start the testing process. In the command string, you specify the measure mode (voltage or current), and the SMU or voltmeter (VM1...VMn) that performs the measurement.

An SMU that is used to measure current is always specified in the trigger command string by its mapped function number (1...8).[28] However, because you can use an SMU to measure voltage either directly (as mapped SMU1...SMU8) or as a mapped voltmeter (VM1...VM8), the SMU is specified in the trigger command string by a unique identifier. For example, a physical SMU that has been mapped as SMU5 (using KCON) is specified by the unique identifier 7. See the command format details below.

After sending the trigger command string, addressing the Model 4200-SCS to talk sends the output data (reading) string to the computer.

27. The assigned “n” value for a voltage source (VSn) or voltmeter (VMn) depends on how instruments are mapped in KCON. Range of possible values: 1-8.
28. The mapped function number normally corresponds to the Model 4200-SCS channel number - the physical SMU number (mapping non-corresponding SMU function numbers, though possible, is not recommended).

Use the following command string to trigger a measurement:

```
AABB
```

Measure mode

AA = TVVoltage measurement
= TI Current measurement

Measure channel

Voltage measure mode specified (AA = TV):[29]

BB
= 1 SMU1
= 2 SMU2
= 3 SMU3
= 4 SMU4
= 5 VM1
= 6 VM2
= 7 SMU5
= 8 SMU6
= 9 SMU7
= 10 SMU8
= 11 VM3
= 12 VM4
= 13 VM5
= 14 VM6
= 15 VM7
= 16 VM8

Current measure mode specified (AA = TI):[29]

B
= 1 SMU1
= 2 SMU2
= 3 SMU3
= 4 SMU4
= 5 SMU5
= 6 SMU6
= 7 SMU7
= 8 SMU8

Output data

After sending the command string to trigger a measurement, and addressing the Model 4200-SCS to talk, the output data string will be sent to the computer in the following format:

```
X Y Z ±N.NNNN E±NN
```

X The status of the data (where X = N for a normal reading)

Y The measure channel (Y = A through F)

Z The measure mode (Z = V or I)

±N.NNNN E±NN is the reading (mantissa and exponent)

29. When channels are mapped to different functions (VM or VS functions), KXCI tries to trigger measurements on the specified channels. However, if the mapped function for a channel does not match the requested measurement, then KXCI reports an error. For example, suppose the mapped function for a channel is VS2, but the requested measurement is TI2. Then KXCI reports an error, because a VS cannot measure current.

Commands common to system and user modes

The following command strings are valid in both the system and user operating modes, and are used for the following operations:

- Set integration time
- Control service request for Data Ready
- Clear data buffer
- Obtain firmware revision levels
- Set global measurement resolution
- Set the lowest current-measurement range
- Auto calibration
- Exit on compliance
- Switch between 4200 and 4145 modes

Set integration time

The integration time is the time it takes to perform a measurement conversion. In generals a short integration time provides the fastest measurement speed at the expense of noise. Conversely, a long integration time provides stable readings at the expense of speed.

- **Preconfigured settings:** KXCI provides three preconfigured integration time settings that are equivalent to the **Fast**, **Normal**, and **Quiet** settings described in Section 6. Integration time is based on power line cycles (PLC). Assuming 60Hz line power, the integration time for a 1 PLC setting is 16.67 msec (1/60).
- **Custom-configured setting**: KXCI also provides a custom integration time setting that combines delay factor, filter factor, and A/D integration time, which is comparable to the individual **Delay Factor**, **Filter Factor**, and **A/D Converter Integration Time** settings described in Section 6. The custom setting is available only in 4200 extended mode.

Use the following command string to set integration time:

```
ITA
```

IT		Command string prefix		
		Integration time (User mode)	Speed (System mode)	
A	= 1	Short (0.1 PLC)	preconfigured selection	Fast
	= 2	Medium (1.0 PLC)	preconfigured selection	Normal
	= 3	Long (10 PLC)	preconfigured selection	Quiet
	= 4,X,Y,Z	Custom-configured setting, where:		
		X = 0.0 to 10.0	Delay factor[30]	
		Y = 0.0 to 10.0	Filter factor[30]	
		Z = 0.01 to 10.0	A/D converter integration time, in number of PLCs.	

Example 1: Preconfigured setting

The following command string sets the integration time to 1.0 PLC:

```
IT2
```

Example 2: Custom-configured setting

The following command string sets the delay factor to 2.5, the filter factor to 0.6, and the A/D converter integration time to 1.3 PLCs.

```
IT4,2.5,0.6,1.3
```

30. For information about these factors, refer to Section 6.

Retrieve instrument ID

The following command string places the ID of the instrument in a particular buffer:

```
ID
```

The instrument ID depends on whether you are in the 4200 mode or whether you are in the 4145 mode. Refer to Table 7-2.

To retrieve the ID value from the buffer, address the instrument to talk. Use of the `enter` command is appropriate for the GPIB card (CIC brand) that is presently used in the Model 4200.

Control service request for "Data Ready"

The GPIB provides a status byte register to monitor instrument operation. Two bits of this register are bit B0 (Data Ready) and bit B6 (RQS). After all programmed measurements are completed, the data becomes ready for output and bit B0 (Data Ready) sets.

If service request for Data Ready is enabled, bit B6 (RQS) will also set when data is ready for output. If service request for Data Ready is disabled, bit B6 will not set when data is ready.

The following command string is used to enable or disable service request for Data Ready:

```
DRA
```

DR = Command string prefix

Service request for data ready

A = 0 Disable service request for data ready
= 1 Enable service request for data ready

The following command string disables service request for data ready:

```
DR0
```

Clear data buffer

The GPIB data buffer can hold up to 4096 readings. The following command string clears all readings from the buffer. It also clears bit B0 (Data Ready) of the status byte.

```
BC
```

Set global measurement resolution

The following command string sets the measurement resolution for all channels:

RS A

RS = Command string prefix

A = Resolution, in number of digits:
3 to 7 digits in 4200 extended mode
3 to 5 digits in 4145 emulation mode

Example

The following command provides full SMU resolution in 4200 extended mode:

```
RS 7
```

Instruct a SMU to go a specified range

The following command string Instructs a SMU to go a specified range immediately, rather than wait until the initiation of a test.

```
RI A,B,C
```

RI = Command string prefix

A = SMU number (1 to 8)

B = Range, permitted values:

100E-9 to 100E-3 amps, 4200-SMU without a preamp
1E-12 to 100E-3 amps, 4200-SMU with a preamp

C = Compliance, permitted values: Compliance, permitted values: 10% to 100% of range.

Example

The following command string instructs SMU1 to go to the 1 mA range and set compliance to 1 mA:

```
RI 1, 1E-3, 1E-3
```

Set the lowest current-measurement range

The following command string sets the lowest current range to be used when measuring:

```
RG A,B
```

RG = Command string prefix

A = SMU number (1 to 8)

B = The lowest auto-ranged[31] range. Permitted values:

- 100E-9 to 100E-3 amps, 4200-SMU without a preamp
- 1E-12 to 100E-3 amps, 4200-SMU with a preamp

- 100E-9 to 1 amps, 4210-SMU without a preamp
- 1E-12 to 1 amps, 4210-SMU with a preamp

Example

The following command string sets the lowest range of SMU2, with a preamp, to 10 pA:

```
RG 2, 10E-12
```

Perform auto calibration

The following command string performs auto calibration of an SMU channel:

```
AC A
```

AC = Command string prefix

31. The default auto-ranged ranges are as follows: 100nA without a PreAmp and 1nA with a PreAmp

A = SMU number (1 to 8)

When this command is sent, auto calibration will be performed on the selected SMU channel. The busy bit in the status register is set so that you can detect when auto calibration is finished. the Model 4200-SCS will not respond to any commands while auto calibration is executing.

Example

The following command string performs auto calibration on SMU channel number 1:

```
AC 1
```

Exit on compliance

The following command will set the condition to exit the test if compliance is reached:

```
EC A
```

EC = Command string prefix

A = 0 OFF (do not exit if compliance is reached)
= 1 ON (exit if compliance is reached)

Example

The following command will enable exit on compliance:

```
EC 1
```

Switch between 4145 and 4200 modes

The following command will switch between 4145 and 4200 modes:

EM A,B
EM = Command string prefix

A = 0 4145 mode
= 1 4200 mode

B = 0 For this session only
= 1 For this and all subsequent sessions (writes to KCON file)

Example

The following selects the 4145 Emulation Mode permanently:

```
EM 0,1
```

4200 extended mode-only commands

Get KXCI configuration of the Model 4200-SCS

The following command string returns a result string that indicates the Model 4200-SCS slot configuration for KXCI:

```
*OPT?
```

When the Model 4200-SCS receives this command, it returns the following configuration string:

```
xxx,xxx,xxx,xxx,xxx,xxx,xxx,xxx
```

This string contains a total of eight sets of characters, each set represented by `xxx`. Each character set represents the configuration of a slot in the Model 4200-SCS, as follows:

- If the corresponding slot contains a channel, then `xxx` = one of the following: `SMUn`, `HPSMUn`, `SMUPAn`, `HPSMUPAn`, `VSn`, or `VMn`. Here, `n`=1,2,..8 is the channel number and:
  - `SMU` = Medium power SMU without a preamp
  - `HPSMU` = High power SMU without a preamp
  - `SMUPA` = medium power SMU with a preamp
  - `HPSMUPA` = high power SMU with a preamp
  - `VS` = voltage source (NOTE: Only 20 V is presently supported).
  - `VM` = voltage meter
- If the corresponding slot does not contain a channel, then `xxx` = `""`.

Ethernet command reference

The Ethernet command set includes all the GPIB commands documented in the GPIB command reference earlier in this section, and adds the following command:

SP: This command allows the user to acquire the GPIB spoll byte while in the Ethernet communication mode.

The SP command is not available for GPIB communication.

SMU default settings

The SMUs can be returned to the power-on default settings by transmitting the DCL (device clear) or SDC (selected device clear) general bus command to the Model 4200-SCS. The power-on default settings for the User Mode are listed in Table 9-6.

The two commands are essentially the same, except that DCL returns all instruments (including the SMUs) connected to the bus to their default settings, while SDC only affects the SMUs (any other instrument on the bus is not affected). Note that the device clear commands do not affect the KXCI configuration settings.

Use either of the following C programming commands (GPIB address 17) to return the SMUs to their power-on default settings:

```
transmit ("UNL LISTEN 17 SDC", &status);  // Reset 4200 only.
transmit ("DCL", &status);   // Reset all instruments.
```

NOTE *The `SDC` and `DCL` commands described above are not text-string commands, nor are any of the other commands in this manual that are sent using the `transmit` function. They are low-level commands that must be sent differently than text-string commands. Do not try to use the* **`transmit`** *function to output text-string commands across the GPIB; use the* **`send`** *function for text-string commands.*

Table 9-6
SMU power-on User Mode default settings

Setting	Default
Range	100 µA
Compliance	105 µA
NRdgs	1
Delay Time	0
Hold Time	0
Wait Time	0
Interval	0
Mode	User Mode

System Mode SMU default Settings

When KXCI mode is started, the SMUs are in a default state that mimics the HP Model 4145B, as shown in Table 9-7 and Table 9-8. This means that each SMU is active and part of the test, whether or not the desired test used all the SMUs in the 4200-SCS chassis.

This may be undesirable, as many tests use a different number of SMUs or SMUs in a different state from the HP4145 default. To avoid the complication of building a KXCI test starting with the default setup (Table 9-6, Table 9-7), use the following commands to remove the SMUs from the setup:

```
DE CH1 CH2 CH3 CH4 VM1 VM2 VS1 VS2
```

The above shows the method for a system with eight SMUs. A system with only four SMUs would not require the last four definitions:

```
DE CH1 CH2 CH3 CH4
```

Table 9-7
Channel Definition (Page DE) KXCI defaults

Instrument	Name		Source		Command
	V	I	Mode	Function	
SMU1	V1	I1	COM	CONSTant	CH
SMU2	V2	I2	I	VAR2	CH
SMU3*	V3	I3	V	VAR1	CH
SMU4*	V4	I4	V	CONSTant	CH
SMU5*	VS1	N/A	V	CONSTant	VS
SMU6*	VS2	N/A	V	CONSTant	VS
SMU7*	VM1	N/A	N/A	N/A	VM
SMU8*	VM2	N/A	N/A	N/A	VM

* If there are less than eight SMUs, only the SMUs in the chassis are programmed to the defaults shown.

Table 9-8
Source Setup (Page SS) KXCI defaults

	VAR1 (Command VR)	VAR2 (Command IP)	
Name	V3	I2	
Sweep mode	Linear	Linear	
Start	0.00 V	20.00 µA	
Stop	1.00 V	N/A	
Step	0.010 V	20.00 µA	
Number of steps	101	5	
Compliance	100.0 mA	2V	
Constant	**Source**	**Compliance**	**Command**
V1 COM	0.00 V	105.0 mA	N/A
V4 V	0.00 V	100.0 mA	VC
VS1 V	0.00 V	N/A	SC
VS2 V	0.00 V	N/A	SC

Output data formats

Data format for system mode readings

For system mode operation, the DO command is used to obtain one or more triggered readings. After sending the DO command string and addressing the Model 4200-SCS to talk, the output data string will be sent to the computer in the following format:

```
X±N.NNNNE±NN,X±N.NNNNE±NN,...X±N.NNNNE±NN
```

Data status[32]

X =	N	Normal
	L	Interval too short
	V	Overflow reading (A/D converter saturated)
	X	Oscillation
	C	This channel in compliance
	T	Other channel in compliance

Reading (mantissa and exponent)[33]

```
±N.NNNN E±NN
```

Data format for user mode readings

For user mode operation, the TI or TV command string is used to trigger and obtain a reading. After sending the TI or TV command string and addressing the Model 4200-SCS to talk, the output data string will be sent to the computer in the following format:

```
XYZ ±N.NNNN E±NN
```

Data status[34]

X =	N	Normal
	L	Interval too short
	V	Overflow reading (A/D converter saturated)
	X	Oscillation
	C	This channel in compliance
	T	Other channel in compliance

Measure channel

		Voltage measure mode specified (Z = V; see below)
Y =	A	SMU1
	B	SMU2
	C	SMU3
	D	SMU4
	E	VM1
	F	VM2
	G	SMU5
	H	SMU6
	I	SMU7

32. The hierarchy for data status is L, V, X, C, T, N.
33. Scientific notation for the reading exponent:
    E+00=10^{0}
    E-03=10^{-3} (m)
    E-06=10^{-6} (μ)
    E-09=10^{-9} (n)
    E-12=10^{-12} (p)
34. The hierarchy for data status is L, V, X, C, T, N.

	J	SMU8
	K	VM3
	L	VM4
	M	VM5
	N	VM6
	O	VM7
	P	VM8
		Current measure mode specified (Z = I; see below)
Y =	A	SMU1
	B	SMU2
	C	SMU3
	D	SMU4
	E	SMU5
	F	SMU6
	G	SMU7
	H	SMU8

Measure mode

Z =	V	Voltage
	I	Current

Reading (mantissa and exponent)[35]

±N.NNNN E±NN

Status byte and serial polling

Status byte

The status byte is an 8-bit register that provides status information on instrument operation. When a particular operating condition occurs, one or more bits of the status byte sets. The status byte register is shown in Figure 9-14.

Figure 9-14
Status Byte Register

Bit	B7	B6	B5	B4	B3	B2	B1	B0
Condition	——	RQS	——	Busy	——	——	Syntax Error	Data Ready
Binary Value	——	0/1	——	0/1	——	——	0/1	0/1
Decimal Weights	——	64	——	16	——	——	2	1

The four used bits of the status byte register are described as follows:

35.Scientific notation for the reading exponent:

E+00	=	10^{0}
E-03	=	10^{-3} (m)
E-06	=	10^{-6} (µ)
E-09	=	10^{-9} (n)
E-12	=	10^{-12} (p)

Bit B0, Data Ready

This bit sets (1) when all measurements are completed and data is ready to be output on the GPIB. Any of the following actions will clear (0) bit B0:

- Clears (0) when the data transfer starts.
- Clears (0) when the BC (buffer clear) command is sent to the Model 4200-SCS.
- Clears (0) when the Model 4200-SCS is serial polled.

Bit B1, Syntax Error

This bit sets (1) when an invalid command string is sent to the Model 4200-SCS. Any of the following actions will clear (0) bit B1:

- Clears (0) when the Model 4200-SCS is serial polled.
- Clears (0) when a device clear command (DCL or SDC command) is sent to the Model 4200-SCS.

Bit B4, Busy

This bit is set (1) while a measurement is being performed. It will clear (0) when the measurement is completed.

Bit B6, RQS (request for service)

This bit sets (1) whenever bit B1 (syntax error) sets. If service request for data ready is enabled (DR1 asserted), bit B6 will set whenever bit B0 (data ready) sets. If service request for data ready is disabled (DR0 asserted), bit B6 will not be affected by bit B0.

Bit B6 remains set until one of the following actions occur:

- Clears (0) when the Model 4200-SCS is serial polled.
- Clears (0) when a device clear command (DCL or SDC command) is sent to the Model 4200-SCS.

NOTE *When bit B6 sets, the SRQ (service request) indicator on the KXCI user interface turns on. It turns off when B6 is cleared.*

Serial polling

The serial poll enable (SPE) and serial poll disable (SPD) general bus command sequence is used to serial poll the Model 4200-SCS. Serial polling reads the status byte. Generally, the serial polling sequence is used by the controller to determine which of several instruments has requested service with the SRQ line. However, the status byte of the Model 4200-SCS may be read to determine when an operating condition has occurred.

If you try to obtain data before all the measurements in a test are completed, a GPIB error will occur. To prevent this, you can use serial polling in a program fragment to monitor the data ready bit (B0) of the status byte. When B0 sets to indicate that data is ready, the program will proceed to obtain the measurement data.

After the source-measure testing process is triggered to start (that is, ME1 sent to start a sweep), the following C language programming statement serial polls the instrument.

```
spoll (addr, &poll, &status);
```

Waiting for SRQ

Instead of serial polling the Model 4200-SCS to detect an SRQ, the service request line can be monitored. When an SRQ occurs, the SRQ line goes true. The following C language programming routine can be used to hold up program execution until an SRQ occurs.

```
send(addr, "DR1", &status);
while(!srq());
```

The first statement enables service request for data ready. The second command holds up program execution until the SRQ (data ready) occurs.

Sample programs

Three sample programs (using the C language) are provided to demonstrate operation over the GPIB. For these programs, the KXCI should be configured as follows:

GPIB address: 17
Delimiter: Comma
EOI: Off

Program 1: VAR1 and VAR2 sweep (system mode)

The following program demonstrates how to program the Model 4200-SCS to perform a VAR1 and VAR2 sweep. It assumes that channels 1 through 4 of the KXCI are configured for the SMU function.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include "ieee_32.h"

#define MAXLEN 2048

void main(void)
{
int addr, status, len;
char recv[MAXLEN];
unsigned char spbyte;

addr = 17;

// Initialize card:
initialize(10, 0);

// Set speed to 0.01 PLC, clear buffer, and
// enable service request for data ready:
send(addr, "IT1 BC DR1”,&status);

// Channel definition for SMU1; constant common:
send(addr, "DE CH1,'VE','IE',3,3", &status);

// Define SMU2 for VAR2 I sweep:
send(addr, "CH2,'VB','IB',2,2", &status);

// Define SMU3 for VAR1 V sweep:
send(addr, "CH3,'VC','IC',1,1", &status);
```

```
// Define SMU 4 to be off:
send(addr, "CH4", &status);

// Define V-sources and V-meters to be off:
send(addr, "VS1;VS2;VM1;VM2", &status);

// Source setup; VAR1 linear sweep from 0 to 1V in 50 mV
// steps, with I-compliance set to 50 mA:
send(addr, "SS VR1,0,1,0.05,50E-3", &status);

// Source setup; VAR2 sweep from 0 uA to 40 uA in 10 uA steps:
send(addr, "IP 10E-6,10E-6,4,3", &status);

// Select list display mode:
send(addr, "SM DM2", &status);

// Trigger start of test:
send(addr, "MD ME1", &status);

// Wait for data ready:
while(!srq());

// Save readings in file named "PROG1":
send(addr, "SV 'D PROG1'", &status);
}
```

Program 2: Basic source-measure (user mode)

The following program demonstrates how to program the Model 4200-SCS to perform a basic source-measure operation. It assumes that channels 1 and 2 of the KXCI are configured for the SMU function. The measured current reading performed by SMU1 (channel 1) is output to the computer.

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream.h>
#include "ieee_32.h"

#define MAXLEN 2048

void main(void)
{
int addr, status, len;
char recv[MAXLEN];

addr = 17;

// Initialize card:
initialize(10, 0);

// Select user mode:
send(addr, "US", &status);

// Set speed to 0.01 PLC, clear buffer, and
```

```
// enable service request for data ready:
send(addr, "IT1 BC DR1”,&status);

// Set SMU1 to source 1.5 V on 20 V range, and set compliance
// to 1mA:
send(addr, "DV1,1, 1.5, 1E-3", &status);

// Set SMU2 to source 2V on 20 V range, and set compliance to 1mA:
send(addr, "DV2,1,2,1E-3", &status);

// Trigger test; measure I using SMU1:
send(addr, "TI1", &status);

// Obtain reading:
enter(recv, MAXLEN, &len, addr, &status);

// Stop SMU outputs:
send(addr, "DV1;DV2", &status);
}
```

Program 3: Retrieving saved data (system mode)

The following program demonstrates how to retrieve readings that are saved in a data file. In Program 1, SMU3 performed 80 measurements. The 80 current readings were then saved in a data file named 'PROG1'.

NOTE *The following program assumes that data file 'PROG1' already exists. This data file is created by Program 1.*

```
#include <stdio.h>
#include <stdlib.h>
#include "ieee_32.h"

#define MAXLEN 2048

void main(void)
{
int addr, status, len;
char recv[MAXLEN];

addr = 17;

// Initialize card:
initialize(10, 0);

// Load data saved in file named “PROG1”:
send(addr, "GT 'D PROG1'", &status);

// Output data; 'IC' is the measure channel (SMU3) used by
// Program 1:
send(addr, "DO 'IC'", &status);

// Obtain data:
enter(recv, MAXLEN, &len, addr, &status);
}
```

GPIB error messages

KXCI error messages and numbers are shown in Table 9-9.

Table 9-9
KXCI error messages

Error No.	Error Message
-999	"IEEE32.DLL GPIB driver is not loaded."
-998	"Unable to initialize shared memory."
-997	"Could not establish communication with console."
-996	"GPIB address not sent as argv[1]."
-995	"GPIB address not in 0<= addr <= 31."
-994	"Could not find configuration file."
-993	"GPIB argument error."
-992	"GPIB command error."
-991	"Illegal setup error."
-990	"Trigger Master card not found."
-989	"Command not valid on this page."
-988	"Instrument not mapped."
-987	"Skipping instrument."
-986	"Unsupported command received."
-985	"Unsupported file format error."
-984	"Could not open specified file."

Pulse generator and scope commands

The KXCI commands to control pulse generator and scope are documented in the following topics:

- Keithley pulse card KXCI commands
- KXCI commands to control scope card

Keithley pulse card KXCI commands

The KXCI commands to control the pulse generator are summarized in Table 9-10. Details on these commands (Pulse card command details) follow the table.

Table 9-10
Keithley pulse card commands

Command	Function	Command String
PD	Set pulse load (output impedance)	PD A, BBB.BBBB
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		BBB.BBBB = Output (load) impedance in ohms: 1.0 to 10 e6 = Default: 50.0 ohms
PG	Set pulse trigger mode	PG A, B, C
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		B = Pulse trigger mode: 0 (Burst), 1 (Continuous) or 2 (Trig Burst)
		C = Number of pulses to output (Burst or Trig Burst): 1 to 2^{32}-1
PH	Halt pulse output	PH A
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
PO	Set pulse output state and mode	PO A, B, C
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Table 9-10 (continued)
Keithley pulse card commands

Command	Function	Command String
		B = Pulse output state: 0 (off) or 1 (on) Default: 0 (off)
		C = Pulse output mode: 0 for Normal or 1 for Complement Default: 0 (Normal)
PS	Reset pulse card	PS A
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
PT	Set pulse timing	PT A, BBB.BBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		BBB.BBBB = Pulse period in seconds (floating point number): 5 V range: 10 e-9 to 1 20 V range: 500 e-9 to 1 Default: 1 e-6
		CCC.CCCC = Pulse width in seconds (floating point number): Fast speed: 10 e-9 to (Period - 10 e-9) Slow speed: 250 e-9 to (Period - 10 e-9) Default: 500 e-9
		DDD.DDDD = Pulse rise time in seconds (floating point number): Fast speed: 10 e-9 to 33 e-3 Slow speed: 50 e-9 to 33 e-3 Default: 100 e-9 NOTE: The minimum transition time for pulse source only (no measurement) on the 40V range is 50 ns for the 4225-PMU and 4220-PGU. The 4200-PG2 and 4205-PG2 have a minimum transition time of 100 ns.
		EEE.EEEE = Pulse fall time in seconds (floating point number): Fast speed: 10 e-9 to 33 e-3 Slow speed: 50 e-9 to 33 e-3 Default: 100 e-9 NOTE: The minimum transition time for pulse source only (no measurement) on the 40V range is 50 ns for the 4225-PMU and 4220-PGU. The 4200-PG2 and 4205-PG2 have a minimum transition time of 100 ns.
PV	Set pulse voltage levels	PV A, BBB.BBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		BBB.BBBB = Pulse high value in volts: Fast speed: -5.0 to +5.0 Slow speed: -20.0 to +20.0 Default: 0.0
		CCC.CCCC = Pulse low value in volts: Fast speed: -5.0 to +5.0 Slow speed: -20.0 to +20.0 Default: 0.0
		DDD.DDDD = Pulse range in volts: 5.0 or 20.0
		EEE.EEEE = Current limit value in amps (range and load dependent): 5 V range: -0.2 to +0.2

Table 9-10 (continued)
Keithley pulse card commands

Command	Function	Command String
		20 V range: -0.8 to +0.8 Default: 0.105 (5 V range)
TO	Set output trigger	TO A, BBB.BBB, C
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		BBB.BBBB = Time delay in seconds: Fast speed: 0.0 to (Period - 10-9) Slow speed: 0.0 to (Period - 10-9) Default: 0.0
		C = Output trigger polarity: 0 for Negative, 1 for Positive Default: 1 (rising edge)
TS	Set trigger source	TS A, B
		A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).
		B = Trigger source: 0 (Software triggered) 1 (External — Initial Trigger Only — Rising edge) 2 (External — Initial Trigger Only — Falling edge) 3 (External — Trigger every Pulse — Rising edge) 4 (External — Trigger every Pulse — Falling edge)

Pulse card command details

NOTE *The following documentation includes the corresponding LPTLIB functions for each KXCI command string. Refer to Section 8 for details on pulsing functions.*

PD command: Set output impedance (pulse load)

Using this command, the DUT impedance of the load (DUT) can be independently set for each channel from 1 Ω to 10 M Ω .

Use the following command string to the output impedance:

```
PD A, BBB.BBBB
```

PD — Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Pulse load in ohms

BBB.BBBB = 1.0 to 10 e6
Default: 50.0

Corresponding LPTLIB function: pulse_load

Example

The following command string sets channel 1 of pulse card 1 for an output impedance of 1 kΩ:

```
PD 1, 1e3
```

PG command: Select the trigger mode

This command selects the trigger mode (Continuous, Burst or Trig Burst) and initiates the start of pulse output or arms the pulse card. The trigger mode setting affects both channels of a pulse card. For example, setting channel 1 of pulse card 1 to Continuous also sets channel 2 to Continuous.

Use the following command string to set the trigger mode:

```
PG A, B, C
```

PG = Command string prefix

Pulse card channel

A=1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Trigger mode

B = 0 Burst mode
= 1 Continuous
= 2 Trig Burst
Default: 1

Pulse count in number of pulses (Burst or Trig Burst mode only)

C = 1 to 232-1
Default: 1

Corresponding LPTLIB functions: pulse_trig, pulse_burst_count

Example

The following command string sets pulse card 1 for the Trig Burst trigger mode and a burst count of 50:

```
PG 1, 2, 50
```

PH command: Stop pulse output

Use the following command string to stop all pulse output from the selected pulse card:

```
PH A
```

PH = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Corresponding LPTLIB functions: pulse_halt

Example

The following command string stops pulse output from pulse card 2:

```
PH 3
```

PO command: Set output state and output mode

Using this command string, the output state (on or off) and mode (normal or complement) can be independently set for each channel.

Use the following command string to the output state and mode:

```
PO A, B, C
```

PO = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Output state

B = 0 Off
= 1 On

Default: 0

Output mode

C = 0 Normal
= 1 Complement

Default: 0

Corresponding LPTLIB function: pulse_output, pulse_output_mode

Example

The following command string turns on channel 1 of pulse card -1 and selects complement pulse output:

```
PO 1, 1, 1
```

PS command: Reset the pulse card

Use the following command to reset both channels of the selected pulse card to its default settings:

```
PS A
```

PS = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Corresponding LPTLIB functions: pulse_init

Example

The following command resets pulse card 2:

```
PS 3
```

PT command: Set pulse timing parameters

This command is used to set pulse period, pulse width, pulse rise time and pulse fall time. The pulse period setting affects both channels of a pulse card. For example, setting channel 1 of pulse card 1 to 100 ms also sets the pulse period of channel 2 to 100 ms. For the other timing parameters (pulse width and rise/fall time), each available channel can have its own unique setting.

Use the following command string to set these parameters:

```
PT A, BBB.BBBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
```

PT = Command string prefix

Pulse card channel

A=1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Pulse period in seconds (floating point number)

BBB.BBBB = 20 e-9 to 1 5 V range
= 250 e-9 to 120 V range
Default: 1e-6

Pulse width in seconds (floating point number)

CCC.CCCC = 10 e-9 to (Period - 10 e-9) Fast speed
= 250 e-9 to (Period - 10 e-9) Slow speed
Default: 500 e-9

Rise time in seconds (floating point number)

DDD.DDDD = 10 e-9 to 33 e-3 Fast speed
= 50 e-9 to 33 e-3 Slow speed[36]
Default: 100 e-9

Fall time in seconds (floating point number)

EEE.EEEE = 10 e-9 to 33 e-3 Fast speed
= 50 e-9 to 33 e-3 Slow speed[36]
Default: 100 e-9

Corresponding LPTLIB functions: pulse_period, pulse_width, pulse_rise, pulse_fall

Example

The following command string sets the pulse period of pulse card 1 to 100µs. It also sets channel 1 for a pulse width of 20µs and rise/fall times for 200 ns:

```
PT 1, 100e-6, 20e-6, 200e-9, 200e-9
```

PV command: Set pulse voltage parameters

Using this command string, pulse high, pulse low, range and current limit can be independently set for each channel of the selected pulse card.

Use the following command string to set these parameters:

```
PV A, BBB.BBBB, CCC.CCCC, DDD.DDDD, EEE.EEEE
```

PV = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Pulse high in volts

BBB.BBBB = -5.0 to +5.0 5 V Range (high speed)
= -20.0 to +20.020 V Range (high voltage)
Default: 0.0

Pulse low in volts

CCC.CCCC = -5.0 to +5.0 5 V Range (high speed)
= -20.0 to +20.020 V Range (high voltage)
Default: 0.0

36. The minimum transition time for pulse source only (no measurement) on the slow speed 40 V range is 50 ns for the Model 4225-PMU and Model 4220-PGU. The Model 4200-PG2 and Model 4205-PG2 have a minimum transition time of 100 ns.

Pulse range in volts

DDD.DDDD = 5 5 V Range
= 20 20 V Range
Default: 5

Current limit in amps (range and load dependent)

EEE.EEEE = -0.2 to +0.2 5 V Range (high speed)
= -0.8 to +0.8 20 V Range (high voltage)
Default: 0.105 5 V range

Corresponding LPTLIB functions: pulse_vhigh, pulse_vlow, pulse_range, pulse_current_limit

Example

For pulse card 1 channel 1, the following command string sets pulse high to +2V, pulse low to -2V, pulse range to 5 V and current limit to 200 mA:

```
PV 1, 2, -2, 5, 200e-3
```

TO command: Set trigger output parameters

This command is used to set pulse delay and trigger polarity. The polarity setting affects both channels of a pulse card. For example, setting channel 1 of pulse card 1 to positive trigger polarity to also sets the trigger polarity of channel 2 to positive. For the trigger delay parameter, each available channel can have its own unique delay setting.

Use the following command string to the pulse delay and trigger polarity:

```
TO A, BBB.BBBB, C
```

TO = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Pulse delay in seconds

BBB.BBB = 0.0 to (Period -10 e-9)
Default: 0.0

Trigger polarity

C = 0 Negative
= 1 Positive
Default: 1 (rising edge)

Corresponding LPTLIB function: pulse_delay, pulse_trig_polarity, pulse_output

Example

The following command string sets the pulse delay for channel 1 of pulse card 1 to 2 µs and selects positive trigger polarity:

```
TO 1, 2 e-6, 1
```

TS command: Set trigger source

This command sets the trigger source that will be used to trigger the-pulse card to start its output. With the Software trigger source selected, the PG command will select the trigger mode (Continuous, Burst, or Trig Burst), and initiate the start of pulse output (see PG command: Select the trigger mode).

NOTE *Since trigger source is a card level setting and NOT a channel setting, using channel 1 or 2 will set the card to the specified source card 1. Similarly, channel 3 or 4 will set the source for card 2.*

With an External trigger source selected, the PG command will select the trigger mode and arm pulse output (see PG command: Select the trigger mode).

Pulse output will start when the required external trigger pulse is applied to the Trigger In connector. There is a trigger-in delay of 560 ns. This is the delay from the trigger-in pulse to the time of the rising edge of the output pulse.

For an “Initial Trigger Only” setting, only the first rising or falling trigger pulse will start and control pulse output.

For a “Trigger per Pulse” setting, rising or falling edge trigger pulses will start and control pulse output. After the initial pulse, the pulse output, either continuous or burst, will be output based on the internal pulse generator clock. If pulse-to-pulse synchronization is required over higher count pulse trains, use a "Trigger per pulse" mode. For details, refer to “External triggering” in Section 11.

Use the following command string to set the trigger source:

```
TS A, B
```

TS = Command string prefix

Pulse card channel

A = 1, 2, ... or n pulse card channel number. The largest permissible value of n equals the number of channels in the system (8 maximum).

Trigger source

B
= 0 Software triggered
= 1 External — Initial Trigger Only — Rising Edge
= 2 External — Initial Trigger Only — Falling Edge
= 3 External — Trigger Every Pulse — Rising Edge
= 4 External — Trigger Every Pulse — Falling Edge

Corresponding LPTLIB function: pulse_trig_source

Example

The following command string sets the trigger source to External – Initial Trigger Only – Rising Edge:

```
TS 1, 1
```

KXCI commands to control scope card

A summary of the KXCI commands for the scope card (Model 4200-SCP2 or 4200-SCP2HR) is shown in the following list. Command details are provided in Table 9-11.

NOTE *Additional information on command programming for the scope card is provided in the supplied ZTEC User's Manual.*

Scope commands list

Advanced Trigger commands

- Output Trigger
- Trigger Holdoff
- Trigger Pulse Width

Calculate commands

- Calc Channel Enable
- Calc FFT
- Calc Function
- Calc Immediate
- Calc Limit Test
- Calc Mask Test
- Calc Time Transform

Configure commands

- Acquisition
- Arm
- Auto Setup
- Channel Enable
- Clock
- Envelope View
- Horizontal
- Trigger
- Vertical

Measurement commands

- Measure Immediate
- Measure Method
- Measure Reference

Operate commands

- Abort
- Arm State Query
- Capture Complete Query
- Capture Waveform
- Soft Arm
- Soft Trigger
- Trigger Event Query
- Trigger Timestamp

Readback commands

- Acquisition Query
- Arm Query
- Calc Channel Query
- Channel Query
- Clock Query
- Envelope View Query
- Horizontal Query
- Trigger Query
- Vertical Query

Readback commands: Advanced Trigger

- Measurement Query
- Output Trigger Query
- Trigger Holdoff Query

Utility commands

- Calibrate
- Close
- Error Description
- Errors
- Initialize
- Initiate
- Operation Complete
- Reset
- Save/Recall State
- Self Test
- Status
- Temperature
- Versions

Waveform commands

- Read Waveform
- Read Waveform (returns data)
- Read Waveform (returns timestamps)
- Store Reference Waveform

Table 9-11
KXCI command strings for scope card

Advanced Trigger commands

Output Trigger, Trigger Holdoff, Trigger Pulse Width

Output Trigger

KXCI command:	**:SCOPE:OUTPUT:TRIGGER triggerOutput, state, source, polarity**
Purpose:	Sets the parameters for the selected output trigger, such as trigger state, trigger source event, and polarity.
Parameters:	**triggerOutput** = 0 PXI TTL1 = 3 PXI TTL4 = 6 PXI TTL7 = 1 PXI TTL2 = 4 PXI TTL5 = 7 PXI TTL8 = 2 PXI TTL3 = 5 PXI TTL6 = 8 External Trigger Selects the output trigger to set up for output. **state** = 0 Inactive = 1 Active Allows the user to enable (Active) or disable (Inactive) the selected output trigger line. **source** = 0 Arm Event = 4 10MHz Reference = 1 Trigger Event = 5 500Hz = 2 Constant State = 6 Pulse = 3 Operation Complete Selects the source for the output trigger. There are 4 events that will cause the trigger output to toggle. The last 3 sources are only valid for the external trigger output. The 10MHz Reference outputs a 10MHz clock. 500Hz is a 500Hz TTL level square wave. Pulse is a 1KHz 10 ns pulse. **polarity** = 0 Negative = 1 Positive Sets the polarity of the selected output trigger. Positive polarity is a positive going pulse while negative polarity is a negative going pulse.
Also see:	Other Advanced Trigger commands, Readback commands: Advanced Trigger
ZTEC function:	SCP2: zt450_output_trigger SCP2HR: zt410_output_trigger
Example:	Set the output trigger parameters: output = External Trigger, state = Active, source = Trigger Event, polarity = Positive: `:SCOPE:OUTPUT:TRIGGER 8, 1, 1, 1`

Return to Scope commands list

Trigger Holdoff

KXCI command:	**:SCOPE:TRIG:HOLDOFF holdoff, eventCount**
Purpose:	Sets the trigger holdoff and event count parameters.
Parameters:	**holdoff** = 0 to 655 (seconds) Resolution: 10 ns for 0 to 655.36 µs 100 ns for 655.36 µs to 6.5536 ms 1 us for 6.5536 ms to 65.536 ms 10 us for 65.536 ms to 655 ms 100 us for 655.36 ms to 6.5536s 1 ms for 6.5536s to 65.536s 10 ms for 65.536s to 655s This sets the time to ignore triggers after receiving a trigger. **eventCount** = 1 to 65535 The number of trigger events that have to happen before the instrument is "triggered."
Also see:	Other Advanced Trigger commands, Readback commands: Advanced Trigger
ZTEC function:	SCP2: zt450_trigger_holdoff SCP2HR: zt410_trigger_holdoff
Example:	Configures the holdoff of even count parameters of the scope: holdoff = 1ms, event count = 1: `:SCOPE:TRIG:HOLDOFF 0, 1`

Return to Scope commands list

Table 9-11 (continued)
KXCI command strings for scope card

Trigger Pulse Width	
KXCI command:	**:SCOPE:TRIG:PULSE:WIDTH source, level, mode, lowerLimit, upperLimit**
Purpose:	Configures the instrument for pulse-width triggering. The instrument can be set up to trigger on a pulse with a width greater than a set limit, less than a set limit, between two limits, or outside of two set limits.
Parameters:	**source** = 0 Channel 1 = 1 Channel 2 Determines the source that will be used to trigger the unit.
	level = Full-scale range value Sets the analog level in volts that is considered high.
	mode = 1 Inside = 2 Outside = 3 Less Than = 4 Greater Than Sets the pulse width mode. Less Than checks that the pulse width is less than the lower limit. Greater Than checks that the pulse width is greater than the upper limit. Inside limits checks that the pulse width is within the two limits. Outside limits checks that the pulse width is outside of the two limits.
	lowerLimit = Value (in seconds) for the lower limit Sets the lower pulse width limit in seconds. 4200-SCP2 Range: N - 65535*N sample intervals Resolution: N sample intervals: N = 8 for 2 GS / s N = 4 for 1 GS/ s N = 2 for 500 MS / s N = 1 for <500 MS / s 4200-SCP2HR Range: 10 ns to 655 s Resolution: 10 ns for 10 ns to 655.36 µs 100 ns for 655.36 µs to 6.5536 ms 1 µs for 6.5536 ms to 65.536 ms 10 µs for 65.536 ms to 655 ms 100 µs for 655.36 ms to 6.5536 s 1 ms for 6.5536 s to 65.536 s 10 ms for 65.536 s to 655 s
	upperLimit = Value (in seconds) for the upper limit Sets the upper pulse width limit in seconds. 4200-SCP2 Range: N - 65535*N sample intervals Resolution: N sample intervals:Resolution: N = 8 for 2 GS / s N = 4 for 1 GS / s N = 2 for 500 MS / s N = 1 for <500 MS / s 4200-SCP2HR Range: 10 ns to 655 s 10 ns for 10 ns to 655.36 µs 100 ns for 655.36 µs to 6.5536 ms 1µs for 6.5536 ms to 65.536 ms 10µs for 65.536 ms to 655 ms 100µs for 655.36 ms to 6.5536 s 1ms for 6.5536 s to 65.536 s 10 ms for 65.536 s to 655 s
Also see:	Other Advanced Trigger commands, Readback commands: Advanced Trigger
ZTEC function:	SCP2: zt450_trigger_pulse_width SCP2HR: zt410_trigger_pulse_width
Example:	Configures pulse width triggering: source = Channel 2, level = 3V, mode = Inside, lower limit = 100µs, upper limit = 200µs: `:SCOPE:TRIG:PULSE:WIDTH 11, 3, 1, 1 e-8, 2 e-8`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Calculate commands

Calc Channel Enable, Calc FFT, Calc Function, Calc Immediate, Calc Limit Test, Calc Mask Test, Calc Time Transform.

Calc Channel Enable

KXCI command:	**:SCOPE:CALC:CHAN:ENABLE channel, state**
Purpose:	Configures the instrument to calculate an FFT.
Parameters:	**channel** = 2 Calc Channel 1 = 3 Calc Channel 2 Enables or Disables one calculation channel. Enable the calculation channel into which you wish to store the result before performing the calculation.
	state = 0 Disable = 1 Enable Enables or Disables the selected channel.
Also see:	Other Calculate commands, Calc Channel Query
ZTEC function:	SCP2: zt450_calc_channel_enable SCP2HR: zt410_calc_channel_enable
Example:	Enable Calc Channel 1: `:SCOPE:CALC:CHAN:ENABLE 2, 1`

Return to Scope commands list

Calc FFT

KXCI command:	**:SCOPE:CALC:FFT calculationChannel, source, window**
Purpose:	Configures the instrument to calculate an FFT.
Parameters:	**calculationChannel** = 2Calc Channel 1 = 3Calc Channel 2 This is where the result of the calculation is stored.
	source=0Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the FFT.
	window = 0 Rectangular = 1 Hamming = 2 Hanning = 3 Blackman Specifies which FFT windowing to use.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_fft SCP2HR: zt410_calculate_fft
Example:	Configure FFT calculation: calc channel = Calc Channel 2, source = Reference Channel 3, window = 0: `:SCOPE:CALC:FFT 3, 6, 0`

Return to Scope commands list

Calc Function

KXCI command:	**:SCOPE:CALC:FUNC calculationChannel, operation, source1, source2, range_andOffsetMode, range, offset**
Purpose:	Configures a calculation channel for simple waveform calculations.
Parameters:	**calculationChannel** = 2 Calc Channel 1 = 3 Calc Channel 2 This is where the result of the calculation is stored.

Table 9-11 (continued)
KXCI command strings for scope card

	operation = 0 Add = 4 Integrate = 1 Absolute Value = 5 Invert = 2 Copy = 6 Multiply = 3 Derivative = 7 Subtract Select the operation to be completed. Add, subtract, and multiply use both sources. Copy, invert, integrate, derivative, and absolute value only use source 1.
	source1 = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the first source to use for the chosen operation.
	source2 = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the second source to use for the chosen operation.
	range_andOffsetMode = 0 Auto = 1 Manual Sets the instrument to automatically determine the range and offset or use manually entered values.
	range = Expected voltage range for the result. Specifies the expected range for the calculation result.
	offset = Expected offset voltage Specifies the DC offset voltage that is represented at vertical center for the selected channel.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_function SCP2HR: zt410_calculate_function
Example:	Configure calc channel: calc channel = Channel 1, operation = Multiply, source1 = Channel 1, source2 = Channel 2, range and offset mode = Auto, range = 5 V, offset = 0 V: `:SCOPE:CALC:FUNC 2, 6, 0, 1, 0, 5.0, 0.0`
Return to Scope commands list	
Calc Immediate	
KXCI command:	**:SCOPE:CALC:IMM channel**
Purpose:	This function causes the instrument to do calculations now based on the calculations setup options. For example, to add channel 1 to channel 2, use :SCOPE:CALC:FUNC to set up the add parameters and then use this function to initiate the calculation.
Parameters:	**channel** = 2 Calc Channel 1 = 3 Calc Channel 2 Selects the calculation channel in which to store the calculation result.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_immediate SCP2HR: zt410_calculate_immediate
Example:	Perform calculations for Channel 1: `:SCOPE:CALC:IMM 2`
Return to Scope commands list	
Calc Limit Test	
KXCI command:	**:SCOPE:CALC:LIM:TEST calculationChannel, source, measurement, lowerLimit, upperLimit, continuously**
Purpose:	Configures the instrument to perform a limit test. A limit test compares, the appropriate measurement on the selected source with an upper and lower limit and generates statistics based on the results. It can be set up to run continuously, or stop when a limit is exceeded.

Table 9-11 (continued)
KXCI command strings for scope card

Parameters:	**calculationChannel** = 2 Calc Channel 1 = 3 Calc Channel 2 This is where the result of the calculation is stored.
	source = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the limit test.
	measurement = 0 AC RMS = 11 Maximum = 22 Rise Preshoot = 1 Amplitude = 12 Minimum = 23 Rise Crossing Time = 2 Average = 13 Middle = 24 Rise Time = 3 DC RMS = 14 -Duty = 25 Time of Maximum = 4 Fall Overshoot = 15 -Width = 26 Time of Minimum = 5 Fall Preshoot = 16 +Duty = 27 Cycle Average = 6 Fall Crossing Time = 17 +Width = 28 Cycle Frequency = 7 Fall Time = 18 Period = 29 Cycle Period = 8 Frequency = 19 Phase = 30 Cycle RMS = 9 High = 20 Peak-to-Peak = 10 Low = 21 Rise Overshoot The above parameter values apply to both the SCP2 and SCP2HR scopes. The following parameters apply only to the SCP2HR: = 31 Precise AC RMS = 35 Signal+Noise+Distortion to Noise+Distortion = 32 Precise DC RMS = 36 Effective Number of Bits = 33 Signal to Noise Ratio = 37 Spurious Free Dynamic Range = 34 Total Harmonic Distortion Selects the type of measurement to check limits on.
	lowerLimit = Value of the lower limit Sets the lower limit for the comparison.
	upperLimit = Value of the upper limit Sets the upper limit for the comparison.
	continuously = 0 Single = 1 Continuously Configures the instrument for continuous limit testing or a single limit test. In Single-shot mode, the instrument will stop when a limit is exceeded and the waveform that exceeded the limit is stored in the selected calc channel.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_limit_test SCP2HR: zt410_calculate_limit_test
Example:	Configure limit test: calc channel = Calc Channel 2, source = Input Channel 2, measurement = AC RMS, lower limit = 1V, upper limit = 5 V, continuously = single. `:SCOPE:CALC:LIM:TEST 3, 1, 0, 1, 5, 0`
	Return to Scope commands list
Calc Mask Test	
KXCI command:	**:SCOPE:CALC:MASK:TEST calculationChannel, source, lowerReference, upperReference, continuous**
Purpose:	Configures the instrument to perform a mask test. A mask test is a special case of the limit test. Instead of using a measurement limit, two reference waveforms are used for a point by point limit test. If any part of the waveform falls outside of the mask, the instrument counts a failure. Use the advanced limit test functions to clear, check events, failures and reports.
Parameters:	**calculationChannel** = 2 Calc Channel 1 = 3 Calc Channel 2 This is where the result of the calculation is stored.

Table 9-11 (continued)
KXCI command strings for scope card

	source = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the limit test.
	lowerReference = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the mask test lower limit.
	upperReference = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the mask test upper limit.
	continuous = 0 No (single) = 1 Yes (continuous) Configures the instrument for continuous mask testing or a single mask test.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_mask_test SCP2HR: zt410_calculate_mask_test
Example:	Configure mask test: calc channel = Calc Channel 2, source = Input Channel 2, lower reference = Reference Channel 1, upper reference = Reference Channel 2, continuously = Single: `:SCOPE:CALC:MASK:TEST 3, 1, 4, 5, 0`
Return to Scope commands list	
Calc Time Transform	
KXCI command:	**:SCOPE:CALC:TIME:TRANS calculationChannel, source, points**
Purpose:	Sets the instrument to do a time transform (smoothing) on a waveform.
Parameters:	**calculationChannel** = 2 Calc Channel 1 = 3 Calc Channel 2 This is where the result of the calculation is stored.
	source = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source to use for the Time Transform.
	points = 2 to 40 Specifies the number of points to use in the transform.
Also see:	Other Calculate commands
ZTEC function:	SCP2: zt450_calculate_time_transform SCP2HR: zt410_calculate_time_transform
Example:	Configure a time transform: calc channel = Calc Channel 2, source = Input Channel 2, points = 20: `:SCOPE:CALC:TIME:TRANS 3, 1, 20`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Configure commands

Acquisition, Arm, Auto Setup, Channel Enable, Clock, Envelope View, Horizontal, Trigger, Vertical

Acquisition

KXCI command:	**:SCOPE:ACQUISITION acquireType, acquireCount, initiateContinuously, triggerMode, equivalentPoints**
Purpose:	Configures the acquisition settings.
Parameters:	**acquireType** = 0 Normal = 1 Average = 2 Envelope = 3 Equivalent Time In Normal mode, a single waveform is captured. In Average mode, multiple captured waveforms are averaged. In Envelope mode, the minimum and maximum values of multiple captured waveforms are used to create an envelope. In Equivalent Time mode, additional buckets are filled through multiple acquisitions.
	acquireCount (SCP2) = 2 to 2048 **acquireCount** (SCP2HR) = 2 to 65535 The acquisition count for repetitive acquisition modes. In Normal mode, this parameter is ignored. In Average mode, this specifies the number of waveforms to be averaged before the acquisition is complete. In Envelope mode, this specifies the number of waveforms for which to capture minimum and maximum values before the acquisition is complete.
	initiateContinuously = 0 Off = 1 On Sets the instrument to initiate continuously. This is usually only used for limit and mask tests.
	triggerMode = 0 Auto = 1 Normal Returns the trigger/sweep mode.
	equivalentPoints = 2 to 100 Returns the equivalent count in equivalent_points per sample_point. For example, if the sample size is 100 points and the equivalent points is 20, the actual waveform size is 100*20 = 2000 pts.
Also see:	Other Configure commands, Acquisition Query
ZTEC function:	SCP2: zt450_acquisition SCP2HR: zt410_acquisition
Example:	Configure acquisitions settings of the scope: acquisition type = Average, count = 25, continuous initiation = On, trigger mode = Auto, equivalent points = 10: `:SCOPE:ACQUISITION 1, 25, 1, 0, 10`

Return to Scope commands list

Arm

KXCI command:	**:SCOPE:ARM armSource, armPolarity**
Purpose:	Configures the arm settings.
Parameters:	**armSource** = 0 Software = 4 PXI TTL3 = 8 PXI TTL7 = 1 PXI TTL0 = 5 PXI TTL4 = 9 PXI Star Trigger = 2 PXI TTL1 = 6 PXI TTL5 = 12 External Trigger = 3 PXI TTL2 = 7 PXI TTL6 = 14 Immediate (disable) Used to select internal or external sample clock source. If set to external, the external clock input should remain between 200MHz and the maximum non-interleaved sample rate.

Table 9-11 (continued)
KXCI command strings for scope card

	armPolarity = 0 Negative = 1 Positive Determines the active state of the selected source. If an arm source is selected and the state of the selected source matches the ARM POLARITY state, the unit will arm. In other words, arm is state driven where triggers are edge driven. The following considerations apply when setting the arm polarity: • POSITIVE state defines the active state as the selected source in its high state. • NEGATIVE state defines the active state as the selected source in its low state.
Also see:	Other Configure commands, Arm Query
ZTEC function:	SCP2: zt450_arm SCP2HR: zt410_arm
Example:	Configure arm settings of the scope: arm source = PXI TTL0, arm polarity = Positive `:SCOPE:ARM 1, 1`
Return to Scope commands list	
Auto Setup	
KXCI command:	**:SCOPE:AUTO:SETUP**
Purpose:	Commands the instrument to autoscale. Autoscale changes the range, offset, coupling, and sample rate based on the input signal(s).
Also see:	Other Configure commands
ZTEC function:	SCP2: zt450_auto_setup SCP2HR: zt410_auto_setup
Example:	Autoscale the scope: `:SCOPE:AUTO:SETUP`
Return to Scope commands list	
Channel Enable	
KXCI command:	**:SCOPE:CHAN:ENABLE channel, state**
Purpose:	Enables or disables one input channel.
Parameters:	**channel** = 0 Input Channel 1 = 1 Input Channel 2 Selects the input channel. **state** = 0 Disable = 1 Enable Enables or Disables the selected channel.
Also see:	Other Configure commands, Channel Query
ZTEC function:	SCP2: zt450_channel_enable SCP2HR: zt410_channel_enable
Example:	Enables Channel 2: `:SCOPE:CHAN:ENABLE 1, 1`
Return to Scope commands list	
Clock	
KXCI command:	**:SCOPE:CLOCK clockSource, referenceSource**
Purpose:	Configures the clock settings.
Parameters:	**clockSource** = 0 Internal = 1 External Used to select internal or external sample clock source. If set to external, the external clock input should remain between 200MHz and the maximum non-interleaved sample rate. **referenceSource** = 0 Local = 4 PXI Backplane Sets the source for the 10MHz reference clock that provides the instrument timebase.
Also see:	Other Configure commands, Clock Query

Table 9-11 (continued)
KXCI command strings for scope card

ZTEC function:	SCP2: zt450_clock SCP2HR: zt410_clock
Example:	Configures clock settings of the scope: clock source = Local, reference source = PXI Backplane: `:SCOPE:CLOCK 0, 0`
	Return to Scope commands list
Envelope View	
KXCI command:	**:SCOPE:ENVELOPE:VIEW view**
Purpose:	Sets the active envelope view. Use this function whenever referencing the envelope to select which view will be used. For example, if you want to perform a calculation on an envelope, first select either the max waveform or min waveform with this function then perform the calculation.
Parameters:	**view** = 0 Min = 1 Max Controls which envelope view to set active. Default view is Max.
Also see:	Other Configure commands, Envelope View Query
ZTEC function:	SCP2: zt450_envelope_view SCP2HR: zt410_envelope_view
Example:	Sets envelope view to Max: `:SCOPE:ENVELOPE:VIEW 1`
	Return to Scope commands list
Horizontal	
KXCI command:	**:SCOPE:HORIZ samplePoints, sampleRate, offsetReference, offsetTime**
Purpose:	Configures the horizontal/timebase settings.
Parameters:	**samplePoints** (SCP2) = 100 to maximum memory (2MB or 32MB) **samplePoints** (SCP2HR) = 100 to 8MS/channel (16MS one channel interleaved) The number of samples in a waveform record. The total number of samples for all channels cannot exceed the memory size.
	sampleRate (SCP2) = 2.5kS/s - 1GS/s in 1, 2.5, 5 steps, 1.25GS/s = 2.5GS/s (1 channel interleaved) **sampleRate** (SCP2HR) = 10 kS/s - 200 MS/s in 1, 2, 2.5, 4, & 5 steps = 400MS/s (1 channel interleaved) Specifies the sample rate in samples/second. An invalid sample rate will be rounded down to the next valid sample rate.
	offsetReference = 0.0 to 1.0 Specifies the position (in per unit) within a record to measure the offset time from.
	offsetTime = 0 to 655 (seconds) Resolution: 10 ns for 0 to 655.36 µs 100 ns for 655.36 µs to 6.5536 ms 1 us for 6.5536 ms to 65.536 ms 10 us for 65.536 ms to 655 ms 100 us for 655.36 ms to 6.5536 s 1 ms for 6.5536 s to 65.536 s 10 ms for 65.536 s to 655 s Specifies the time between the trigger and the offset reference in seconds. Because the offset time is always positive, pre-trigger sampling is only possible by setting the reference close to 100 and the offset time to 0.
Also see:	Other Configure commands, Horizontal Query

Table 9-11 (continued)
KXCI command strings for scope card

ZTEC function:	SCP2: zt450_horizontal SCP2HR: zt410_horizontal
Example:	Configures horizontal/timebase settings of the scope: points = 200, rate = 10 kS/s, offset reference = 0, offset time = 0 s: `:SCOPE:HORIZ 200, 10e3, 0, 0`
Return to Scope commands list	
Trigger	
KXCI command:	**:SCOPE:TRIG triggerSource, level, slope**
Purpose:	Configures the edge trigger settings.
Parameters:	**triggerSource** = 0 Software = 5 PXI TTL4 = 10 CH1 Trigger = 1 PXI TTL0 = 6 PXI TTL5 = 11 CH2 Trigger = 2 PXI TTL1 = 7 PXI TTL6 = 12 External Trigger = 3 PXI TTL2 = 8 PXI TTL7 = 13 Pattern Trigger = 4 PXI TTL3 = 9 PXI Star Trigger Determines the source that will be used to trigger the unit.
	level = Full-scale range value Sets the analog level in volts to trigger at. This is only used for Channel 1 or Channel 2 as the trigger source.
	slope = 0 Falling = 1 Rising Sets the active edge of the selected trigger for the instrument to rising edge or falling edge.
Also see:	Other Configure commands, Trigger Query
ZTEC function:	SCP2: zt450_trigger SCP2HR: zt410_trigger
Example:	Configure edge trigger settings of the scope: trigger source = PXI TTL0, level = 5 V, slope = Rising edge: `:SCOPE:TRIG 1, 5, 1`
Return to Scope commands list	
Vertical	
KXCI command:	**:SCOPE:VERT channel, range, offset, coupling, impedance, lowpassFilter, attenuation**
Purpose:	Configures the vertical settings for the selected channel.
Parameters:	**channel** = 0 Channel 1 = 1 Channel 2 Selects the input channel.
	range = full-scale range value (see below) Ranges: 50 Ω: 50 mVpp, 0.1Vpp, 0.25 Vpp, 0.5 Vpp, 1Vpp, 2 Vpp, 5 Vpp, 10 Vpp 1 M Ω:0.1 Vpp, 0.2 Vpp, 0.5 Vpp, 1 Vpp, 2.5 Vpp, 5 Vpp, 10 Vpp, 20 Vpp, 50 Vpp, 100 Vpp Specifies the full scale acquisition range in volts for the specified input channel.
	offset = ±Full-scale range value Specifies the DC offset voltage that is represented at vertical center for the selected channel.
	coupling = 0 AC = 1 DC Sets the input coupling for the selected channel. AC high pass filter settings: • 200 kHz highpass with 50 Ω impedance • 10 Hz highpass with 1 M Ω impedance
	impedance = 50 50 Ω = 1 e6 1 M Ω Sets the input impedance for the selected channel.

Table 9-11 (continued)
KXCI command strings for scope card

	lowpassFilter = 0 Bypass = 1 20MHz Sets the status of the lowpass filter for the selected channel. Note: The high resolution scope card (SCP2HR) does not have the low-pass filter circuit. However, this parameter must be included in the function even though it is a "no-op" (no operation). KXCI will ignore the parameter value. If the scope card is Model 4200-SCP2, this parameter will set the low-pass filter.
	attenuation = 0.9 to 1000 Sets the probe attenuation scale factor (ratio) for the selected channel (0.9:1 to 1000:1).
Also see:	Other Configure commands, Vertical Query
ZTEC function:	SCP2: zt450_vertical SCP2HR: zt410_vertical
Example:	Configures vertical settings of the scope: channel = Channel 1, range = 1Vpp, offset = 0 V, coupling = AC, impedance = 50Ω, filter = Bypass, attenuation = 2:1: `:SCOPE:VERT 0, 1, 0, 0, 50, 0, 2`
Return to Scope commands list	

Measurement commands

Measure Immediate, Measure Method, Measure Reference

Measure Immediate

KXCI command:	**:SCOPE:MEAS:IMM? measurement, source**
Purpose:	Function sends a measurement command and gets the result back.
Parameters:	**measurement** = 0 AC RMS, = 1 Amplitude, = 2 Average, = 3 DC RMS, = 4 Fall Overshoot, = 5 Fall Preshoot, = 6 Fall Crossing Time, = 7 Fall Time, = 8 Frequency, = 9 High, = 10 Low, = 11 Maximum, = 12 Minimum, = 13 Middle, = 14 -Duty, = 15 -Width, = 16 +Duty, = 17 +Width, = 18 Period, = 19 Phase, = 20 Peak-to-Peak, = 21 Rise Overshoot, = 22 Rise Preshoot, = 23 Rise Crossing Time, = 24 Rise Time, = 25 Time of Maximum, = 26 Time of Minimum, = 27 Cycle Average, = 28 Cycle Frequency, = 29 Cycle Period, = 30 Cycle RMS, = 31* Precise AC RMS, = 32* Precise DC RMS * Parameter values 31 and 32 apply only to the SCP2HR. Selects the type of measurement to perform. **source** = 0 Input Channel 1, = 1 Input Channel 2, = 2 Calc Channel 1, = 3 Calc Channel 2, = 4 Reference Channel 1, = 5 Reference Channel 2, = 6 Reference Channel 3, = 7 Reference Channel 4 Selects the source to use for the measurement.
Returned results:	**result** = Measured value for the selected source. Returns the result of the measurement.
Also see:	Other Measurement commands, Waveform commands
ZTEC function:	SCP2: zt450_measure_immeditate SCP2HR: zt410_measure_immeditate
Example:	Perform a measurement and return the result: measurement = Peak-to-Peak, source = Input Channel 1: `:SCOPE:MEAS:IMM? 20, 0`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Measure Method	
KXCI command:	**:SCOPE:MEAS:METH method, gateType, gateStart, gateStop, edgeNumber**
Purpose:	Configures the method to use for measurements.
Parameters:	**method** = 0 Entire Waveform = 1 Gated Selects the method to use for the measurement. Entire Waveform uses the entire record. Gated only uses a section of the waveform determined by the gate points/times.
	gateType = 0 Time = 1 Points This control determines if the gate start and stop are interpreted in seconds or sample points. It is only used if the method type is gated.
	gateStart = Time or points value Specifies the start of the gate in seconds or points.
	gateStop = Time or points value Specifies the end of the gate in seconds or points.
	edgeNumber = Time or points value Selects the edge to perform the measurement on (when appropriate).
Also see:	Other Measurement commands, Waveform commands
ZTEC function:	SCP2: zt450_measure_method SCP2HR: zt410_measure_method
Example:	Configure measure method for entire waveform: `:SCOPE:MEAS:METH 0`
Return to Scope commands list	

Measure Reference	
KXCI command:	**:SCOPE:MEAS:REF referenceMethod, lowReference, midReference, highReference**
Purpose:	Sets the upper and lower threshold levels for measurements.
Parameters:	**referenceMethod** = 0 Percent = 1 Voltage Set the reference levels in absolute voltages or relative percentages.
	lowReference = Value in volts or percent) Set the low reference level in volts or percent.
	midReference = Value in volts or percent) Set the mid reference level in volts or percent.
	highReference = Value in volts or percent) Set the mid reference level in volts or percent.
Also see:	Other Measurement commands, Waveform commands
ZTEC function:	SCP2: zt450_measure_reference SCP2HR: zt410_measure_reference
Example:	Configure threshold levels for measurements: method = Voltage, low reference = 0 V, mid reference = 2.5 V, high reference = 5 V: `:SCOPE:MEAS:REF 1, 0, 2.5, 5`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Operate commands

Abort, *Arm State Query*, *Capture Complete Query*, *Capture Waveform*, *Soft Arm*, *Soft Trigger*, *Trigger Event Query*, *Trigger Timestamp*

Abort

KXCI command:	**:SCOPE:ABORT**
Purpose:	Terminates all ongoing processes and returns the unit to the idle state. Data resulting from the ongoing processes may be corrupt.
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_abort SCP2HR: zt410_abort
Example:	Abort all processes and return to idle: `:SCOPE:ABORT`

Return to Scope commands list

Arm State Query

KXCI command:	**:SCOPE:ARM:STATE?**
Purpose:	Queries the Waiting for Arm bit of the Operation Register.
Returned values:	**state** - State of the Waiting for Arm bit of the Operation Register: 0 = Not Waiting for Arm 1 = Waiting for Arm
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_state_query SCP2HR: zt410_state_query
Example:	Acquire arm state: `:SCOPE:ARM:STATE?`

Return to Scope commands list

Capture Complete Query

KXCI command:	**:SCOPE:CAPTURE:COMPLETE?**
Purpose:	Performs an Event Status Register Query.
Returned values:	**state** - Returns whether a capture is complete. Used in Asynchronous capture mode: 0 = Not Complete 1 = Complete
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_capture_complete_query SCP2HR: zt410_capture_complete_query
Example:	Query the Event Status Register: `:SCOPE:CAPTURE:COMPLETE?`

Return to Scope commands list

Capture Waveform

KXCI command:	**:SCOPE:CAPTURE:WAVEFORM time-out**
Purpose:	Initiates the instrument and captures a waveform. Normal mode will wait until the waveform is captured. Asynchronous mode may exit before the waveform is captured. Use ztXXX_capture_complete_query() to check if the capture is complete.
Parameters:	**time-out** = 0 Asynchronous = ≥65535 Infinite Sets the capture function time-out in seconds. Infinite waits until the waveform is complete (that is, until the next trigger edge is detected, before returning). Asynchronous returns immediately. Use the capture complete query to verify completion.
Also see:	Other Operate commands

Table 9-11 (continued)
KXCI command strings for scope card

ZTEC function:	SCP2: zt450_capture_waveform SCP2HR: zt410_capture_waveform
Example:	Capture a waveform immediately: `:SCOPE:CAPTURE:WAVEFORM 0`
Return to Scope commands list	
Soft Arm	
KXCI command:	**:SCOPE:SOFT:ARM armState**
Purpose:	Arms or disarms the unit through software. If arm source is set to software, then this function is used to either arm or disarm the unit. When armed the unit will begin trigger detection. When disarmed, the unit will ignore triggers.
Parameters:	**armState** = 0 Disarm = 1 Arm Software arm control to arm or disarm the instrument.
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_soft_arm SCP2HR: zt410_soft_arm
Example:	Arm the scope: `:SCOPE:SOFT:ARM 1`
Return to Scope commands list	
Soft Trigger	
KXCI command:	**:SCOPE:SOFT:TRIG**
Purpose:	Causes an immediate software trigger event regardless of trigger source setting. If enabled, the trigger outputs onto the PXI backplane will also toggle when a trigger manual is executed.
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_soft_trigger SCP2HR: zt410_soft_trigger
Example:	Trigger the scope: `:SCOPE:SOFT:TRIG`
Return to Scope commands list	
Trigger Event Query	
KXCI command:	**:SCOPE:TRIG:EVENT?**
Purpose:	Queries the Trigger Event Bit of the Operation Status Register (OSR).
Returned values:	**state** - State of the Trigger Event bit of the Operation Register: 0 = No Trigger Event 1 = Trigger Event Occurred
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_trigger_event_query SCP2HR: zt410_trigger_event_query
Example:	Query the Trigger Event Bit of the OSR: `:SCOPE:TRIG:EVENT?`
Return to Scope commands list	
Trigger Timestamp	
KXCI command:	**:SCOPE:TRIG:TIMESTAMP?**
Purpose:	Acquires the timestamp from the last trigger. Returns the timestamp value in seconds (0 to 1).
Returned values:	**time** - Timestamp from the last trigger (in seconds): 0 to 1
Also see:	Other Operate commands
ZTEC function:	SCP2: zt450_trigger_timestamp SCP2HR: zt410_trigger_timestamp

Table 9-11 (continued)
KXCI command strings for scope card

Example:	Query timestamp: `:SCOPE:TRIG:TIMESTAMP?`
Return to Scope commands list	

Readback commands

Acquisition Query, Arm Query, Calc Channel Query, Channel Query, Clock Query, Envelope View Query, Horizontal Query, Trigger Query, Vertical Query

Acquisition Query

KXCI command:	**:SCOPE:ACQUISITION?**
Purpose:	Queries the acquisition settings of the oscilloscope.
Returned values:	**acquireType** - Type of acquisition: 0 = Normal 1 = Average 2 = Envelope 3 = Equivalent Time
	acquireCount - Acquisition count for repetitive acquisition modes: SCP2:2 to 2048 SCP2HR: 2 to 65535
	initiateContinuously - State of continuous initiation: 0 = Off 1 = On
	triggerMode -Trigger/sweep mode: 0 = Auto 1 = Normal
	equivalentPoints - Equivalent count in equivalent_points per sample_point. 2 to 100
Also see:	Other Readback commands, Acquisition
ZTEC function:	SCP2: zt450_acquisition_query SCP2HR: zt410_acquisition_query
Example:	Query the acquisition settings: `:SCOPE:ACQUISITION?`
Return to Scope commands list	

Arm Query

KXCI command:	**:SCOPE:ARM?**
Purpose:	Query the arm settings.
Returned values:	**armSource -** Source used to arm the scope: 0 = Software, 4 = PXI TTL3, 8 = PXI TTL7 1 = PXI TTL0, 5 = PXI TTL4, 9 = PXI Star Trigger 2 = PXI TTL1, 6 = PXI TTL5, 12 = External Trigger 3 = PXI TTL2, 7 = PXI TTL6, 14 = Immediate (disable)
	armPolarity - Active state of the selected source: 0 = Negative 1 = Positive
Also see:	Other Readback commands, Arm
ZTEC function:	SCP2: zt450_arm_query SCP2HR: zt410_arm_query
Example:	Query arm settings: `:SCOPE:ARM?`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Calc Channel Query	
KXCI command:	**:SCOPE:CALC:CHAN? channel**
Purpose:	Queries and returns the state of the selected calc channel.
Parameters:	**channel** = 2 Calc Channel 1 = 3 Calc Channel 2 Selects the calc channel.
Returned values:	**state** - State of the selected channel: 0 = Disabled 1 = Enabled
Also see:	Other Readback commands, Calc Channel Enable
ZTEC function:	SCP2: zt450_calc_channel_query SCP2HR: zt410_calc_channel_query
Example:	Query Calc Channel 2: `:SCOPE:CALC:CHAN? 3`
Return to Scope commands list	

Channel Query	
KXCI command:	**:SCOPE:CHAN? channel**
Purpose:	Queries and returns the state of the selected input channel.
Parameters:	**channel** = 0 Input Channel 1 = 1 Input Channel 2 Selects the input channel.
Returned values:	**state** - State of the selected channel: 0 = Disabled 1 = Enabled
Also see:	Other Readback commands, Channel Enable
ZTEC function:	SCP2: zt450_channel_query SCP2HR: zt410_channel_query
Example:	Query Input Channel 2: `:SCOPE:CHAN? 1`
Return to Scope commands list	

Clock Query	
KXCI command:	**:SCOPE:CLOCK?**
Purpose:	Queries the clock settings.
Returned values:	**clockSource** - Clock source: 0 = Internal 1 = External
	referenceSource - Source for the 10 MHz reference clock that provides the instrument timebase: 0 = Local 4 = PXI Backplane
Also see:	Other Readback commands, Clock
ZTEC function:	SCP2: zt450_clock_query SCP2HR: zt410_clock_query
Example:	Query the clock settings: `:SCOPE:CLOCK?`
Return to Scope commands list	

Envelope View Query	
KXCI command:	**:SCOPE:ENVELOPE:VIEW?**
Purpose:	Query the active envelope view.

Table 9-11 (continued)
KXCI command strings for scope card

Returned values:	**view** - Active envelope view: 0 = Minimum 1 = Maximum
Also see:	Other Readback commands, Envelope View
ZTEC function:	SCP2: zt450_envelope_view_query SCP2HR: zt410_envelope_view_query
Example:	Query the envelope view: `:SCOPE:ENVELOPE:VIEW?`
	Return to Scope commands list
Horizontal Query	
KXCI command:	**:SCOPE:HORIZ?**
Purpose:	Queries the instrument for the horizontal and timebase settings.
Returned values:	**samplePoints** - Number of samples in a waveform record: SCP2: 100 to maximum memory (2MB or 32MB). The total number of samples for all channels cannot exceed the memory size. SCP2HR: 100 to 64MS/channel (128MS if only one channel enabled)
	sampleRate - Sample rate in samples/second: SCP2: 2.5kS/s to 1GS/s in 1, 2.5, 5 steps, 1.25GS/s 2.5GS/s (1 channel interleaved) SCP2HR: 20 kS/s to 200MS/s in 1, 2, 2.5, 4, & 5 steps 400MS/s (1 channel interleaved)
	offsetReference - Position (in per unit) within a record to measure the offset time from: SCP2: 0 to 100% SCP2HR: 0 to 1.0
Also see:	Other Readback commands, Horizontal
ZTEC function:	SCP2: zt450_horizontal_query SCP2HR: zt410_horizontal_query
Example:	Query the horizontal and timebase settings: `:SCOPE:HORIZ?`
	Return to Scope commands list
Trigger Query	
KXCI command:	**:SCOPE:TRIG?**
Purpose:	Query the edge trigger settings.
Returned values:	**triggerSource -** Source used to trigger the scope: 0 = Software, 1 = PXI TTL0, 2 = PXI TTL1, 3 = PXI TTL2, 4 = PXI TTL3, 5 = PXI TTL4, 6 = PXI TTL5, 7 = PXI TTL6, 8 = PXI TTL7, 9 = PXI Star Trigger, 10 = CH1 Trigger, 11 = CH2 Trigger, 12 = External Trigger, 13 = Pattern Trigger
	level - Analog trigger level (in volts). This is only used for Channel 1 or Channel 2 as the trigger source: Full-scale range value
	slope - Activating edge of the selected trigger: 0 = Falling 1 = Rising
	mode - Trigger mode: 0 = Edge Triggering, 1 = Pulse Width Inside Limits, 2 = Pulse Width Outside Limits, 3 = Pulse Width Less Than, 4 = Pulse Width Greater Than, 5 = Video Triggering
Also see:	Other Readback commands, Trigger
ZTEC function:	SCP2: zt450_trigger_query SCP2HR: zt410_trigger_query

Table 9-11 (continued)
KXCI command strings for scope card

Example:	Query arm settings: `:SCOPE:TRIG?`
Return to Scope commands list	
Vertical Query	
KXCI command:	**:SCOPE:VERT? channel**
Purpose:	Queries the vertical settings for the selected channel. The Returned values are for the selected input channel.
Parameters:	**channel** = 0 Input Channel 1 = 1 Input Channel 2 Selects the input channel.
Returned values:	**range** - Full-scale acquisition range (in volts).
	offset - DC offset voltage that is represented at vertical center.
	coupling - Input coupling: 0 = AC 1 = DC
	impedance - Input impedance: 50 = 50 Ω 1 e6 = 1 M Ω
	lowpassFilter - Lowpass filter: SCP2: 0 = Bypass 1 = 20MHz Note: If the high-resolution scope card (Model 4200-SCP2HR) is installed, the returned value for will be "0" (off). The high-resolution scope card does not have the low-pass filter.
	attenuation - Probe attenuation scale factor: 0.9 to 1000
Also see:	Other Readback commands, Vertical
ZTEC function:	SCP2: zt450_vertical_query SCP2HR: zt410_vertical_query
Example:	Query the vertical settings for Input Channel 1: `:SCOPE:VERT? 0`
Return to Scope commands list	

Readback commands: Advanced Trigger

Measurement Query, Output Trigger Query, Output Trigger Query

Measurement Query	
KXCI command:	**:SCOPE:MEAS?**
Purpose:	Queries the measurement settings
Returned values:	**method** - Method used for the measurement. Entire Waveform uses the entire record. Gated only uses a section of the waveform determined by the gate points/times: 0 = Entire 1 = Gated
	gateStartTime - Start of the gate in seconds.
	gateStopTime - End of the gate in seconds.
	gateStartPoints - Start of the gate in points.
	gateStopPoints - End of the gate in points.
	edgeNumber - Edge used for a measurement.
	referenceMethod - Reference levels in absolute voltages or relative percentages: 0 = Percent 1 = Voltage
	lowReference - Returns the low reference level in volts or per unit.

Table 9-11 (continued)
KXCI command strings for scope card

	midReference - Returns the mid reference level in volts or per unit.
	highReference - Returns the high reference level in volts or per unit.
Also see:	Other Readback commands: Advanced Trigger, Measure Method, Measure Reference
ZTEC function:	SCP2: zt450_measurement_query SCP2HR: zt410_measurement_query
Example:	Query the measurement settings: `:SCOPE:MEAS?`
Return to Scope commands list	
Output Trigger Query	
KXCI command:	**:SCOPE:OUTPUT:TRIGGER? triggerOutput**
Purpose:	Queries the parameters for the selected output trigger.
Parameters:	**triggerOutput** = 0 PXI TTL0 = 3 PXI TTL3 = 6 PXI TTL6 = 1 PXI TTL1 = 4 PXI TTL4 = 7 PXI TTL7 = 2 PXI TTL2 = 5 PXI TTL5 = 8 External Trigger Selects the output trigger to set up for output.
Returned values:	**state** - State of the selected output trigger: 0 = Inactive 1 = Active
	source - Event that causes an output trigger: 0 = Arm Event 1 = Trigger Event 2 = Constant State 3 = Operation Complete
	polarity - Polarity of the selected output trigger: 0 = Negative 1 = Positive
Also see:	Other Readback commands: Advanced Trigger, Output Trigger
ZTEC function:	SCP2: zt450_output_trigger_query SCP2HR: zt410_output_trigger_query
Example:	Query the parameters for the PXI TTL1 output trigger: `:SCOPE:OUTPUT:TRIGGER? 1`
Return to Scope commands list	
Trigger Holdoff Query	
KXCI command:	**:SCOPE:TRIG:HOLDOFF?**
Purpose:	Queries the trigger holdoff and event count parameters.
s:	**holdoff -** Time to ignore triggers after receiving a trigger: 0 to 655 (seconds)
	eventCount - Number of events that have to happen before a trigger occurs: 1 to 65535
Also see:	Other Readback commands: Advanced Trigger, Trigger Holdoff
ZTEC function:	SCP2: zt450_trigger_holdoff_query SCP2HR: zt410_trigger_holdoff_query
Example:	Query trigger holdoff and event count parameters: `:SCOPE:TRIG:HOLDOFF?`
Return to Scope commands list	

Utility commands

Calibrate, Close, Error Description, Errors, Initialize, Initiate, Operation Complete, Reset, Save/Recall State, Self Test, Status, Temperature, Versions

Table 9-11 (continued)
KXCI command strings for scope card

Calibrate	
KXCI command:	**:SCOPE:CALIBRATE**
Purpose:	Performs an internal self calibration routine on the instrument and returns the result. Note that the two input channels must be disconnected before starting the calibration. This function resets the time-out value to infinite before starting the calibration and resets it to the default value when completed. Make sure that the instrument isn't interrupted during calibration. The calibration tables could be corrupted.
Returned values:	**Result** - Result of the internal self-calibration: 0 = Successful
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_calibrate SCP2HR: zt410_calibrate
Example:	Calibrate the scope: `:SCOPE:CALIBRATE`
Return to Scope commands list	
Close	
KXCI command:	**:SCOPE:CLOSE**
Purpose:	Closes the VISA communication session opened by Initialize.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_close SCP2HR: zt410_close
Example:	Close communication session: `:SCOPE:CLOSE`
Return to Scope commands list	
Error Description	
KXCI command:	**:SCOPE:ERROR:DESCR? code**
Purpose:	This functions accepts a single error code and returns a string that describes the error. Note: These are instrument error codes not VISA error codes.
Parameters:	**code** = Error code to be described.
Returned values:	**description** - The description of the error code. Max length = 128 characters.
Also see:	Errors, other Utility commands
ZTEC function:	SCP2: zt450_error_description SCP2HR: zt410_error_description
Example:	Return description of error -901: `:SCOPE:ERROR:DESCR? -901`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Errors	
KXCI command:	**:SCOPE:ERRORS?**
Purpose:	Returns the number of errors and an array containing the error numbers.
Returned values:	**number_ofErrors** - Returns number of errors in queue: 0 to 32
	errors - Returns all entries in the error log and clears the error log. Multiple errors are stored sequentially in the error log with the oldest error first.
Also see:	Error Description, other Utility commands
ZTEC function:	SCP2: zt450_errors SCP2HR: zt410_errors
Example:	Return errors: `:SCOPE:ERRORS?`
Return to Scope commands list	
Initialize	
KXCI command:	**:SCOPE:INITIALIZE**
Purpose:	Sets up the VISA session and establishes communications with the instrument. An instrument reset may also be selected with this function call. The initialize routine is automatically performed when KXCI is started. This command only needs to be used if the VISA session was closed by the :SCOPE:CLOSE command (see "Close").
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_initialize SCP2HR: zt410_initialize
Example:	Initialize the scope: `:SCOPE:INITIALIZE`
Initiate	
KXCI command:	**:SCOPE:INITIATE**
Purpose:	Initiates the instrument. While initiated, the instrument is enabled to acquire waveforms and perform calculations and measurements.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_initiate SCP2HR: zt410_initiate
Example:	Initiate the scope: `:SCOPE:INITIATE`
Return to Scope commands list	
Operation Complete	
KXCI command:	**:SCOPE:OPER:COMPLETE?**
Purpose:	Sends an operation complete query to the instrument.
Returned values:	**queryResult** - Operation Complete Query: 0 = Operation Not Complete 1 = Operation Complete
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_operation_complete SCP2HR: zt410_operation_complete
Example:	Query Operation Complete: `:SCOPE:OPER:COMPLETE?`
Return to Scope commands list	

Table 9-11 (continued)
KXCI command strings for scope card

Reset	
KXCI command:	**:SCOPE:RESET**
Purpose:	Performs a hardware reset function that returns the instrument to the initial default condition.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_reset SCP2HR: zt410_reset
Example:	Reset the scope: `:SCOPE:RESET`
Return to Scope commands list	
Save/Recall State	
KXCI command:	**:SCOPE:SAVE:RECALL state, stateNumber**
Purpose:	Stores the current state of the instrument to the selected storage index in nonvolatile memory or returns the state of the instrument from a stored condition.
Parameters:	**state** = 0 Save = 1 Recall Controls if a state should be stored or recalled.
	stateNumber (SCP2) = 1 to 50 **stateNumber** (SCP2HR) = 1 to 31 Instrument storage index (location) control.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_save_recall_state SCP2HR: zt410_save_recall_state
Example:	Save scope state in location 10: `:SCOPE:SAVE:RECALL 0, 10`
Return to Scope commands list	
Self Test	
KXCI command:	**:SCOPE:SELFTEST**
Purpose:	Initiates an instrument self test and returns the test status register.
Returned values:	**selfTestStatus** - Returns the present condition of the Test Status Register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 6 is returned, the binary equivalent is 0000000000000110. This binary value indicates that Bits 1 and 2 are set. The SRAM Test and ROM Test has failed. *Test Status Register*: Bit 0 - Baseboard Test Failed Bit Bit 1 - SRAM Test Failed Bit Bit 2 - ROM Test Failed Bit Bit 3 - Unused Bit 4 - Reference Oscillator Test Failed Bit Bit 5 - DRAM Test Failed Bit Bit 6 - Flash Memory Test Failed Bit Bit 7 - Unused Bit Bit 8 - Input 1-2 Register Test Failed Bit Bit 9 - Input 1 RAM Test Failed Bit Bit 10 - Input 2 RAM Test Failed Bit Bit 11 - PLL Test Failed Bit Bit 12-15 - Unused
Also see:	Other Utility commands

Table 9-11 (continued)
KXCI command strings for scope card

ZTEC function:	SCP2: zt450_self_test SCP2HR: zt410_self_test
Example:	Perform scope self-test: `:SCOPE:SELFTEST`
Return to Scope commands list	
Status	
KXCI command:	**:SCOPE:STATUS? registerSelect**
Purpose:	Returns the current state of all status event or condition registers.
Parameters:	**registerSelect** = 0: Condition Register = 1: Event Register Selects whether to download the status condition or event registers. Condition registers indicate the current state of the instrument. Event registers indicate states that occurred since the last event check.
Returned values:	**statusRegister** - Returns the present condition of the status register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 20 is returned, the binary equivalent is 0000000000010100. This binary value indicates that Bits 2 and 4 are set. *Status Register*: Bit 0 - Unused Bit Bit 1 - Unused Bit Bit 2 - Error Log Not Empty Bit Bit 3 - Questionable Summary Bit Bit 4 - Message Available Bit Bit 5 - Standard Event Summary Bit Bit 6 - Master Summary Bit Bit 7 - Operation Summary Bit
	frequencyRegister - Returns the present condition of the frequency register: Returns a 0 if the PLL is locked or a 1 if the PLL is unclocked. *Frequency Register*: Bit 0 - PLL Unlocked Bit Bits 1-7 - Unused Bits
	testRegister - Returns the present condition of the Test Status Register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 6 is returned, the binary equivalent is 0000000000000110. This binary value indicates that Bits 1 and 2 are set. The SRAM Test and ROM Test has failed. *Test Status Register*: Bit 0 - Baseboard Test Failed Bit Bit 1 - SRAM Test Failed Bit Bit 2 - ROM Test Failed Bit Bit 3 - Unused Bit 4 - Reference Oscillator Test Failed Bit Bit 5 - Unused Bit Bit 6 - Flash Memory Test Failed Bit Bit 7 - Unused Bit Bit 8 - Input 1-2 Register Test Failed Bit Bit 9 - Input 1 RAM Test Failed Bit Bit 10 - Input 2 RAM Test Failed Bit Bit 11 - PLL Test Failed Bit Bit 12-15 - Unused

Table 9-11 (continued)
KXCI command strings for scope card

	operationRegister - Returns the present condition of the operation register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 20 is returned, the binary equivalent is 0000000000010100. This binary value indicates that Bits 2 and 4 are set. *Operation Register*: Bit 0 - Calibrating Bit Bit 1 - Settling Bit Bit 2 - Ranging Bit Bit 3 - Sweeping Bit Bit 4 - Measuring Bit Bit 5 - Waiting for Trigger Bit Bit 6 - Waiting for Arm Bit Bit 7 - Unused Bit Bit 8 - Trigger Event Bit Bit 9 - Data Capture Event Bit Bit 10 - Limit Test Event Bit Bit 11 - Autodownload Event Bit Bit 12-15 - Unused Bits
	standardRegister - Returns the present condition of the standard register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 20 is returned, the binary equivalent is 0000000000010100. This binary value indicates that Bits 2 and 4 are set. *Standard Register*: Bit 0 - Operation Complete Bit Bit 1 - Request Control Bit Bit 2 - Query Error Bit Bit 3 - Device Dependent Error Bit Bit 4 - Execution Error Bit Bit 5 - Command Error Bit Bit 6 - User Request Bit Bit 7 - Power On Bit
	questionableRegister - Returns the present condition of the questionable register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 32 is returned, the binary equivalent is 0000000000100000. This binary value indicates that Bit 5 is set. *Questionable Register*: Bit 0 - Voltage Register Bit Bit 1-3 - Unused Bits Bit 4 - Temperature Bit Bit 5 - Frequency Register Bit Bit 6-7 - Unused Bits Bit 8 - Calibration Register Bit Bit 9 - Test Register Bit Bit 10-15 - Unused Bits
	voltageRegister - Returns the present condition of the voltage register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 2 is returned, the binary equivalent is 0000000000000010. This binary value indicates that Bit 1 is set. *Voltage Register* (SCP2): *Voltage Register* (SCP2HR): Bit 0 - Input 1 Overload Bit Bit 4 - Input 1 Overvoltage Bit Bit 1 - Input 2 Overload Bit Bit 5 - Input 2 Overvoltage Bit Bit 2-3, and Bit 6-7 - Unused Bits

Table 9-11 (continued)
KXCI command strings for scope card

	calibrationRegister - Returns the present condition of the calibration register: Returns a decimal value. The binary equivalent of the decimal value indicates which bits of the Status Test Register are set (1) or clear (0). For example, if decimal value 2 is returned, the binary equivalent is 0000000000000010. This binary value indicates that Bit 1 is set. Bit 0 - Calibration Storage Failed Bit Bit 1 - Offset Zero Failed Bit Bit 2 - Offset Scale Failed Bit Bit 3 - Null Balance Failed Bit Bit 4 - Gain Balance Failed Bit Bit 5 - Sample Rate Null Failed Bit Bit 6-7 - Unused Bits
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_status SCP2HR: zt410_status
Example:	Query event registers: `:SCOPE:STATUS? 1`
Return to Scope commands list	
Temperature	
KXCI command:	**:SCOPE:TEMP?**
Purpose:	Query the temperature of the scope.
Returned values:	**temperature** - Temperature of scope in °C. A temperature of 60° C or higher will cause a temperature error.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_temperaure SCP2HR: zt410_temperaure
Example:	Query the temperature of the scope: `:SCOPE:TEMP?`
Return to Scope commands list	
Versions	
KXCI command:	**:SCOPE:VERSIONS?**
Purpose:	Returns the ID string, driver revision and configuration versions.
Returned values:	**ID** - Instrument identification including manufacturer, model number, serial number and firmware version as a block of ASCII string data up to 44 characters in length. The string will be in the form: ZTEC,ZT450PXI,S/N nnn,Version n.nn
	driver_rev - Version of the C (CVI) driver string in the form: Rev x.xx, dd/mm/yy, CVI n.n
	configuration - Instrument identification as an array where the first three elements are DSP firmware version, baseboard FPGA version and module FPGA version.
Also see:	Other Utility commands
ZTEC function:	SCP2: zt450_versions SCP2HR: zt410_versions
Example:	Query the versions data: `:SCOPE:VERSIONS?`
Return to Scope commands list	
Waveform commands *Read Waveform, Read Waveform (returns data), Read Waveform (returns timestamps), Store Reference Waveform*	
Read Waveform	
KXCI command:	**:SCOPE:READ:WAVEFORM source**

Table 9-11 (continued)
KXCI command strings for scope card

Purpose:	Data is captured for the source channel with this command and can be read with the `:SCOPE:READ:WAVEFORM:DATA?` command.
Parameters:	**source** = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source of the waveform to be read.
Also see:	Other Waveform commands, Measurement commands
ZTEC function:	SCP2: zt450_read_waveform SCP2HR: zt410_read_waveform
Example:	Reads waveform for Input Channel 2: `:SCOPE:READ:WAVEFORM 1`
Return to Scope commands list	
Read Waveform (returns data)	
KXCI command:	**:SCOPE:READ:WAVEFORM:DATA?**
Purpose:	Returns an array of voltage waveform data samples.
Also see:	Other Waveform commands, Measurement commands
ZTEC function:	SCP2: zt450_read_waveform SCP2HR: zt410_read_waveform
Example:	Returns waveform data: `:SCOPE:READ:WAVEFORM:DATA?`
Return to Scope commands list	
Read Waveform (returns timestamps)	
KXCI command:	**:SCOPE:READ:WAVEFORM:TIMESTAMPS?**
Purpose:	Returns an array of timestamps that correspond to the waveform data.
Also see:	Other Waveform commands, Measurement commands
ZTEC function:	SCP2: zt450_read_waveform SCP2HR: zt410_read_waveform
Example:	Returns timestamps: `:SCOPE:READ:WAVEFORM:TIMESTAMPS?`
Return to Scope commands list	
Store Reference Waveform	
KXCI command:	**:SCOPE:STORE:REF:WAVEFORM referenceChannel, source**
Purpose:	Stores a waveform from one of the sources in a reference waveform channel.
Parameters:	**referenceChannel** = 4 Reference Channel 1 = 5 Reference Channel 2 = 6 Reference Channel 3 = 7 Reference Channel 4 Selects the reference channel to store the waveform in.
	source = 0 Input Channel 1 = 4 Reference Channel 1 = 1 Input Channel 2 = 5 Reference Channel 2 = 2 Calc Channel 1 = 6 Reference Channel 3 = 3 Calc Channel 2 = 7 Reference Channel 4 Selects the source of the waveform to be read.
Also see:	Other Waveform commands, Measurement commands
ZTEC function:	SCP2: zt450_store_reference_waveform SCP2HR: zt410_store_reference_waveform
Example:	Stores a waveform from Input Channel 1 into Reference Channel 1: `:SCOPE:STORE:REF:WAVEFORM 0, 4`
Return to Scope commands list	

Scope error codes

The following scope error codes apply to both the Models 4200-SCP2HR and 4200-SCP2:

```
{0, -100, "ERROR!! -100: Command error"},
{0, -101, "ERROR!! -101: Invalid character"},
{0, -102, "ERROR!! -102: Syntax error"},
{0, -103, "ERROR!! -103: Invalid separator"},
{0, -104, "ERROR!! -104: Data type error"},
{0, -105, "ERROR!! -105: Get not allowed"},
{0, -108, "ERROR!! -108: Parameter not allowed"},
{0, -109, "ERROR!! -109: Missing parameter"},
{0, -110, "ERROR!! -110: Command header error"},
{0, -111, "ERROR!! -111: Header separator error"},
{0, -112, "ERROR!! -112: Mnemonic too long"},
{0, -113, "ERROR!! -113: Undefined header"},
{0, -114, "ERROR!! -114: Header suffix out-of-range"},
{0, -118, "ERROR!! -118: Query not allowed"},
{0, -120, "ERROR!! -120: Numeric data error"},
{0, -121, "ERROR!! -121: Invalid char in number"},
{0, -123, "ERROR!! -123: Exponent too large"},
{0, -124, "ERROR!! -124: Too many digits"},
{0, -128, "ERROR!! -128: Numeric data not allowed"},
{0, -130, "ERROR!! -130: Suffix error"},
{0, -131, "ERROR!! -131: Invalid suffix"},
{0, -134, "ERROR!! -134: Suffix too long "},
{0, -138, "ERROR!! -138: Suffix not allowed"},
{0, -140, "ERROR!! -140: Character data error"},
{0, -141, "ERROR!! -141: Invalid character data"},
{0, -144, "ERROR!! -144: Character data too long"},
{0, -148, "ERROR!! -148: Character data not allowed"},
{0, -150, "ERROR!! -150: String data error"},
{0, -151, "ERROR!! -151: Invalid string data"},
{0, -158, "ERROR!! -158: String data not allowed"},
{0, -160, "ERROR!! -160: Block data error"},
{0, -161, "ERROR!! -161: Invalid block data"},
{0, -168, "ERROR!! -168: Block data not allowed"},
{0, -170, "ERROR!! -170: Expression error"},
{0, -171, "ERROR!! -171: Invalid expression"},
{0, -178, "ERROR!! -178: Expression data not allowed"},
{0, -180, "ERROR!! -180: Macro error"},
{0, -181, "ERROR!! -181: Invalid outside macro"},
{0, -183, "ERROR!! -183: Invalid inside macro"},
{0, -184, "ERROR!! -184: Macro parameter error"},
{0, -200, "ERROR!! -200: Execution error"},
{0, -201, "ERROR!! -201: Invalid while in local"},
{0, -202, "ERROR!! -202: Settings lost due to RTL"},
{0, -203, "ERROR!! -203: Command protected"},
```

```
{0, -210, "ERROR!! -210: Trigger Error"},
{0, -211, "ERROR!! -211: Not ready for trigger"},
{0, -212, "ERROR!! -212: Not ready for arm"},
{0, -213, "ERROR!! -213: Already initiated"},
{0, -214, "ERROR!! -214: Not ready for trigger"},
{0, -220, "ERROR!! -220: Parameter error"},
{0, -221, "ERROR!! -221: Settings conflict"},
{0, -222, "ERROR!! -222: Data out of range"},
{0, -223, "ERROR!! -223: Too much data"},
{0, -224, "ERROR!! -224: Illegal parameter value"},
{0, -225, "ERROR!! -225: Out of memory"},
{0, -226, "ERROR!! -226: Lists not the same length"},
{0, -230, "ERROR!! -230: Data corrupt or stale"},
{0, -231, "ERROR!! -231: Questionable data"},
{0, -232, "ERROR!! -232: Data has invalid format"},
{0, -233, "ERROR!! -233: Incompatible version"},
{0, -240, "ERROR!! -240: Hardware error"},
{0, -241, "ERROR!! -241: Hardware missing"},
{0, -250, "ERROR!! -250: Mass storage error"},
{0, -251, "ERROR!! -251: Missing mass storage"},
{0, -252, "ERROR!! -252: Missing media"},
{0, -253, "ERROR!! -253: Corrupt media"},
{0, -254, "ERROR!! -254: Media full"},
{0, -255, "ERROR!! -255: Directory full"},
{0, -256, "ERROR!! -256: File name not found"},
{0, -257, "ERROR!! -257: File name error"},
{0, -258, "ERROR!! -258: Media protected"},
{0, -260, "ERROR!! -260: Expression execution failed"},
{0, -261, "ERROR!! -261: Math expression execution failed"},
{0, -270, "ERROR!! -270: Macro execution error"},
{0, -271, "ERROR!! -271: Macro syntax error"},
{0, -272, "ERROR!! -272: Macro execution error"},
{0, -273, "ERROR!! -273: Illegal macro label"},
{0, -274, "ERROR!! -274: Macro parameter error"},
{0, -275, "ERROR!! -275: Macro definition too long"},
{0, -276, "ERROR!! -276: Macro recursion error"},
{0, -277, "ERROR!! -277: Macro redefinition not allowed"},
{0, -278, "ERROR!! -278: Macro header not found"},
{0, -280, "ERROR!! -280: Program error"},
{0, -281, "ERROR!! -281: Can not create program"},
{0, -282, "ERROR!! -282: Illegal program name"},
{0, -283, "ERROR!! -283: Illegal variable name"},
{0, -284, "ERROR!! -284: Program currently running"},
{0, -285, "ERROR!! -285: Program syntax error"},
{0, -286, "ERROR!! -286: Program runtime error"},
{0, -290, "ERROR!! -290: Memory usage error"},
{0, -291, "ERROR!! -291: Out of memory"},
```

```
{0, -292, "ERROR!! -292: Reference name does not exist"},
{0, -293, "ERROR!! -293:  Reference name already exists"},
{0, -294, "ERROR!! -294:  Incompatible Type"},
{0, -300, "ERROR!! -300:  Device specific error"},
{0, -310, "ERROR!! -310:  System error"},
{0, -311, "ERROR!! -311:  Memory error"},
{0, -312, "ERROR!! -312:  PUD memory lost"},
{0, -313, "ERROR!! -313:  Calibration memory corrupted"},
{0, -314, "ERROR!! -314:  Configuration memory corrupted"},
{0, -315, "ERROR!! -315:  Manufacturing info corrupted"},
{0, -320, "ERROR!! -320:  Storage Fault"},
{0, -321, "ERROR!! -321:  Out of memory for an internal operation"},
{0, -330, "ERROR!! -330:  Self test failed"},
{0, -340, "ERROR!! -340:  Calibration failed"},
{0, -350, "ERROR!! -350:  Queue overflow"},
{0, -360, "ERROR!! -360:  Communications error "},
{0, -361, "ERROR!! -361:  Parity error in program message"},
{0, -362, "ERROR!! -362:  Framing error in program message"},
{0, -363, "ERROR!! -363:  Input buffer overrun"},
{0, -400, "ERROR!! -400:  Query error"},
{0, -410, "ERROR!! -410:  Query interrupt error"},
{0, -420, "ERROR!! -420:  Query un-terminated error"},
{0, -430, "ERROR!! -430:  Query deadlock error"},
{0, -440, "ERROR!! -440:  Query un-terminated after indefinite
response"},
{0, -500, "ERROR!! -500:  Power on"},
{0, -700, "ERROR!! -700:  Request control"},
{0, -800, "ERROR!! -800:  Operation complete"},
{0, -1001, "ERROR!! -1001:  PLL unlocked"},
{0, -1002, "ERROR!! -1002:  Boot Failed"},
{0, -1003, "ERROR!! -1003:  Wave Invalid"},
{0, -1004, "ERROR!! -1004:  Overtemp"},
{1, 0, "Unknown Error Code"},
```

KXCI CVU commands

There are two command modes used to control the CVU using KXCI: User mode and System mode. This mimics the paradigm used to control the SMUs in I/V tests.

User mode is analogous to LPTLIB (when running local) in that all of the commands are processed sequentially, and the user has full control over the execution sequence and flow control. In this mode, when a measurement is complete, only one point is generated (R + Jx, Cp, Gp, Cs, Rs, and so on).

System mode is similar to KITE ITMs in that all CVU test parameters are buffered and executed automatically by the system when the user sends the run command. Full sweeps are executed by the CVU card itself, and the output sequencing, settling, and sweeping are all handled internally. The user simply returns the entire buffer of test data.

It is important to note that many of the commands below are valid in both User and System mode. When in user mode, the command is processed immediately and sent directly to the card. In system mode, the parameter is buffered in a test object, and does not take effect until the execute command is sent. At that time, the CVU's firmware acts upon the settings and executes the desired test sequence.

NOTE *All of the commands act on the selected card which is set with the :CVU:CHANNEL command.*

Table 9-12
KXCI CVU command strings

KXCI CVU commands ***User Mode, System Mode, Modeless***	
User Mode	
KXCI command:	**:CVU:MEASZ?**
Purpose:	Triggers and returns single Z-measurement using current CVU settings. When the command is complete, the single reading is available over GPIB or Ethernet.
System Mode	
KXCI command:	**:CVU:SOAK:DCV voltage**
Purpose:	Sets the presoak DC voltage for CVU card for ALL sweeps (that is, frequency sweep, DC voltage sweep, AC voltage sweep, and so on) for the selected CVU card.
Parameters:	**voltage:** Voltage to bias before test sequence begins. Voltages range from -30 V to +30 V.
KXCI command:	**:CVU:BIAS:DCV:SAMPLE biasv, samples**
Purpose:	Configures the CVU to bias a DC voltage and sample *n* Z-measurements for the CVU card. The other parameters are set by their respective commands. *n* represents the integer value passed to the samples parameter.
Parameters:	**biasv:** Voltage to source while sampling Z-measurements. Voltages range from -30 V to +30 V.
	samples: The number of Z-measurements the CVU will make for the test operation. Valid values range from 1 to 4096.
KXCI command:	**:CVU:SWEEP:FREQ fstart, fstop, <order>**
Purpose:	Configures the CVU to sweep frequency and sample Z-measurements for the selected CVU card. Only the start frequency, stop frequency, and sweep order are set here. The other parameters are set by their respective commands.
Parameters:	**fstart:** The frequency that is used for capturing the initial sample in the sweep.
	fstop: The frequency that is used for capturing the final sample in the sweep.
	order: <optional parameter> Determines whether the test will bias DC voltage and sweep frequency, or step voltage and sweep frequency. 1: The voltage will just bias the voltage set by the :CVU:DCV command. This will also be the behavior when the parameter is omitted. 2: The voltage will step based on the vstart, vstop, vstep parameters set by the :CVU:SWEEP:DCV command. ***NOTE*** *Values are coerced to one of the 28 discrete frequencies or 37 discrete frequencies for the 4200-CVU and 4210-CVU, respectively.*
KXCI command:	**:CVU:SWEEP:DCV dcvstart, dcvstop, dcvstep**
Purpose:	Configures the CVU to sweep DC voltage and sample Z-measurements for the selected CVU card.
Parameters:	**dcvstart:** start voltage
	dcvstop: stop voltage
	dcvstep: voltage step size ***NOTE*** *The other parameters (AC voltage level, frequency, and so on) are set by their respective commands. Valid voltage levels are -30 V to +30 V.*
KXCI command:	**:CVU:SWEEP:ACV acvstart, acvstop, acvstep**

Table 9-12 (continued)
KXCI CVU command strings

Purpose:	Configures the CVU to sweep AC voltage and sample Z-measurements for the selected CVU card.
Parameters:	**acvstart:** start AC voltage
	acvstop: stop AC voltage
	acvstep: AC voltage step size ***NOTE*** *Valid AC voltage levels are 10 mV to 100 mV. The other parameters (DC voltage level, frequency, and so on) are set by their respective commands.*
KXCI command:	**:CVU:DELAY:STEP stepd**
Purpose:	Sets the hold time for the CVU test on the selected card for the :CVU:SWEEP:FREQ and :CVU:SWEEP:DCV operations.
Parameters:	**stepd:** Hold time to apply the pre-soak value (set by the :CVU:SOAK:DCV command). Valid values range from 0 to 999 seconds.
KXCI command:	**:CVU:DELAY:SWEEP sweepd**
Purpose:	Sets the sweep delay for the CVU test on the selected card.
Parameters:	**sweepd:** Used when in sweeping mode (which is all sweep types except :CVU:BIAS:DCV:SAMPLE). Delay is in seconds. Valid values are 0 to 999 seconds.
KXCI command:	**:CVU:SAMPLE:HOLDT holdt**
Purpose:	Sets the hold time for a sampling mode test on the selected card.
Parameters:	holdt: This is only used when executing when executing the :CVU:BIAS:DCV:SAMPLE command. Value is in seconds. Valid values are 0 to 999 seconds.
KXCI command:	**:CVU:SAMPLE:INTERVAL interval**
Purpose:	Sets the delay between samples for the selected card.
Parameters:	interval: This is only used when executing when executing the :CVU:BIAS:DCV:SAMPLE command. Value is in seconds. Valid values are 0 to 999 seconds.
KXCI command:	**:CVU:SWEEP:LISTDCV pt1, pt2, pt3…..ptn**
Purpose:	Configures the selected CVU card to sweep arbitrary DC voltage points and sample Z-measurements. Only starting voltage sweep points are set here.
Parameters:	**pt1, pt2, pt3…..ptn:** The other parameters (AC voltage level, frequency, and so on) are set by their respective commands. There can be up to 4096 sweep points specified. ***NOTE*** *Valid voltage levels are -30 V to +30 V.*
KXCI command:	**:CVU:TEST:RUN**
Purpose:	Start a CVU test on specified card. Use the serial poll byte to determine when not busy or data ready just like SMU I/V tests.
KXCI Command:	**:CVU:TEST:ABORT**
Purpose:	Stops a running KXCI CVU test. This system mode command terminates all ongoing processes and returns the unit to the idle state. Data resulting from the ongoing processes may be corrupt. Note that this command is not appropriate to stop a CVU KXCI test running from the Remote UTM mode (for more information, refer to “Calling KULT user libraries remotely” on page 9-102). This command is not valid in User Mode.
KXCI command:	**:CVU:STANDBY state**
Purpose:	Configure the selected CVU card to disable DC bias at the end of a test, or leave it active.

Table 9-12 (continued)
KXCI CVU command strings

Parameters:	**state:** 1 **<TRUE>** disables the DCV output 0 **<FALSE>** leaves it active. (that is, do not auto standby)
KXCI command:	**:CVU:DATA:Z?** **:CVU:DATA:VOLT?** **:CVU:DATA:FREQ?** **:CVU:DATA:STATUS?** **:CVU:DATA:TSTAMP?**
Purpose:	Queries the respective measurements of the CVU once a test is complete. ***NOTE*** *Z-measurements are returned as semi-colon delimiter pairs. Each reading is then delimited with commas.*
Modeless Commands	
KXCI command:	**:CVU:CHANNEL chan**
Purpose:	Selects CVU card on which subsequent CVU commands will act. The majority of systems will only have 1 CVU card, but this command allows for the user to configure multiple cards.
Parameters:	**chan:** The default is 1, so at the start, all CVU KXCI commands will configure channel 1 unless specified otherwise.
KXCI command:	**:CVU:MODEL model**
Purpose:	Set measurement model for selected CVU card.
Parameters:	**model:** 0 Z, theta 1 R + jx 2 Cp, Gp 3 Cs, Rs 4 Cp, D 5 Cs, D
KXCI command:	**:CVU:MODE mode**
Purpose:	Sets User or System mode.
Parameters:	**mode:** 0 User Mode 1 System Mode
KXCI command:	**:CVU:RESET**
Purpose:	Send soft reset to specified card. Places the card in default state.
Parameters:	DC Voltage: 0 V — Model: R + jx AC voltage: 30 mV — Range: 1 mA Freq: 100 kHz
KXCI command:	**:CVU:SPEED speed,<delay factor>,<filter factor>,<aperture>**
Purpose:	Set the measurement speed for the selected CVU card.

Table 9-12 (continued)
KXCI CVU command strings

<table>
<tr><td>Parameters:</td><td>speed: Used to apply one of four defined speed selections.
<table><tr><td>0</td><td>Fast</td></tr><tr><td>1</td><td>Normal</td></tr><tr><td>2</td><td>Quiet</td></tr><tr><td>3</td><td>Custom</td></tr></table></td></tr>
<tr><td></td><td>delay factor: 0 to 100</td></tr>
<tr><td></td><td>filter factor: 0 to 100</td></tr>
<tr><td></td><td>aperture: 0.006 to 10.002 PLCS

NOTE The last 3 parameters are only used if custom speed is selected.</td></tr>
<tr><td>KXCI command:</td><td>:CVU:ACV aclevel</td></tr>
<tr><td>Purpose:</td><td>Sets the AC drive level for specified CVU card.</td></tr>
<tr><td>Parameters:</td><td>aclevel: If in user mode, this takes immediate effect.
If in system mode, the value is buffered for use in all sweeps except AC Voltage sweep (that is set with the :CVU:SWEEP:ACV command).

NOTE Valid range: 10 mV to 100 mV.</td></tr>
<tr><td>KXCI command:</td><td>:CVU:DCV dclevel</td></tr>
<tr><td>Purpose:</td><td>Sets the DC bias voltage for specified CVU card.</td></tr>
<tr><td>Parameters:</td><td>dclevel: If in user mode, this takes immediate effect.
If in system mode, the value is buffered for use in Frequency and AC Voltage sweeps as well as DC Bias/Sample.

NOTE Valid voltage levels are -30 V to +30 V.</td></tr>
<tr><td>KXCI command:</td><td>:CVU:ACZ:RANGE range</td></tr>
<tr><td>Purpose:</td><td>Sets the AC measurement range for the specified CVU card.</td></tr>
<tr><td>Parameters:</td><td>range:<table><tr><td>0</td><td>Auto</td></tr><tr><td>1 E-6</td><td>1 μA</td></tr><tr><td>30 E-6</td><td>30 μA</td></tr><tr><td>1 E-3</td><td>1 mA</td></tr></table></td></tr>
<tr><td>KXCI command:</td><td>:CVU:FREQ freq</td></tr>
<tr><td>Purpose:</td><td>Sets the frequency for the AC source for the specified CVU card.</td></tr>
</table>

Table 9-12 (continued)
KXCI CVU command strings

Parameters:	**freq:** If in user mode, this takes immediate effect. If in system mode, the value is buffered for use in Voltage Bias/Sample tests, List and linear Voltage sweeps, and AC Voltage sweeps. ***NOTE*** *Valid range: 10 mV to 100 mV. Values that fall between supported freq. will be coerced to closest supported freq.*
KXCI command:	**:CVU:LENGTH len**
Purpose:	Selects the cable length for the specified CVU card.
Parameters:	**len:** 0 — 0 m 1.5 — 1.5 m 3.0 — 3.0 m
KXCI command:	**:CVU:CORRECT open, short, load**
Purpose:	Enables/disables open, short, and load correction for the specified CVU card.
Parameters:	**open:** 0 / 1 0 = off / 1 = on **short:** 0 / 1 0 = off / 1 = on **load:** 0 / 1 0 = off / 1 = on ***NOTE*** *Each of the parameters is a Boolean (0 = false, 1 = true).*
KXCI command:	**:CVU:DCV:OFFSET offsetv**
Purpose:	Apply offset value to the DC low terminal.
Parameters:	**offsetv:** Voltage offset to apply while sampling Z-measurements. Voltages range from -30 V to +30 V.
KXCI command:	**:CVU:CONFIG:ACVHI source**
Purpose:	Allow the user to define the source terminal (AC only) for the CVU test to be performed. Unless set otherwise, the default AC source terminal is HCUR/HPOT.
Parameters:	**source**: The source terminal to be used. 1 - HCUR/HPOT 2 - LCUR/LPOT
KXCI command:	**:CVU:CONFIG:DCVHI source**
Purpose:	Allow the user to define the source terminal (DC only) for the CVU test to be performed. Unless set otherwise, the default DC source terminal is HCUR/HPOT.
Parameters:	**source**: The source terminal to be used. 1 - HCUR/HPOT 2 - LCUR/LPOT

Code examples

Example1

The following code segment sets CVU1 to perform a system mode sweep of DC voltage from 5 V to 10 V in 1V steps. After the test completes, the Z, DCV, F, timestamps, and status values are queried. The forcing function is then changed to perform a List Sweep of DC voltage (2V, 4V, 3V, 5 V, 7V) and the test is run again with all other test conditions as previous.

```
// Set CVU to System Mode
send(addr, ":CVU:MODE 1", &status);

// Soft Reset card
send(addr, ":CVU:RESET", &status);

// Set measurement model to Z, theta
send(addr, ":CVU:MODEL 0", &status);

// Set speed to Normal
send(addr, ":CVU:SPEED 1", &status);

// Set AC drive level to 45 mV at 7MHz
send(addr, ":CVU:ACV 0.045", &status);
send(addr, ":CVU:FREQ 7E+6", &status);

// Set DC bias level to 10 V
send(addr, ":CVU:DCV 10", &status);

// Select 1mA measurement range
send(addr, ":CVU:ACZ:RANGE 1E-3", &status);

// Turn off open/short/load compensation an set
// cable len to 1.5 m
send(addr, ":CVU:CORRECT 0,0,0", &status);
send(addr, ":CVU:LENGTH 1.5", &status);

// Set test function to DC Voltage sweep from 5 to 10 V
send(addr, ":CVU:SWEEP:DCV 5,10,1", &status);

// Set 1s delay between points
send(addr, ":CVU:DELAY:SWEEP 1.0", &status);

// Start the test
send(addr, ":CVU:TEST:RUN", &status);

// Monitor the spoll byte for test completion
WaitForTestCompletion();

// Query all the data
send(addr, ":CVU:DATA:Z?", &status);
enter(recvstr, MAXLEN, &len, addr, &status);

send(addr, ":CVU:DATA:VOLT?", &status);
enter(recvstr, MAXLEN, &len, addr, &status);

send(addr, ":CVU:DATA:FREQ?", &status);
```

```
enter(recvstr, MAXLEN, &len, addr, &status);

send(addr, ":CVU:DATA:TSTAMP?", &status);
enter(recvstr, MAXLEN, &len, addr, &status);

send(addr, ":CVU:DATA:STATUS?", &status);
enter(recvstr, MAXLEN, &len, addr, &status);

// Change sweep mode to List Sweep
send(addr, ":CVU:SWEEP:LISTDCV 2,4,3,5,7", &status);

// And rerun the test
send(addr, ":CVU:TEST:RUN", &status);
WaitForTestCompletion();

send(addr, ":CVU:DATA:VOLT?", &status);
enter(recvstr, MAXLEN, &len, addr, &status);
```

Example 2

The next example shows how to perform a single measurement in User mode. The program sets up CVU1 to output 30 mV@10 Mhz AC. The DC voltage is set to 5 V. Once single measurement is retrieved, the AC source is set to 60 mV@5 MHz and a new reading is acquired.

```
// Set CVU to User Mode
send(addr, ":CVU:MODE 0", &status);

// Soft Reset card
send(addr, ":CVU:RESET", &status);

// Set measurement model to Cs,Rs
send(addr, ":CVU:MODEL 3", &status);

// Set speed to Quiet
send(addr, ":CVU:SPEED 2", &status);

// Set AC drive level to 30 mV at 10MHz
send(addr, ":CVU:ACV 0.030", &status);
send(addr, ":CVU:FREQ 10E+6", &status);

// Set DC bias level to 5 V
send(addr, ":CVU:DCV 5", &status);

// Select 30 uA measurement range
send(addr, ":CVU:ACZ:RANGE 30E-6", &status);

   // Trigger a new measurement, and retrieve it
send(addr, ":CVU:MEASZ", &status);
enter(recvstr, MAXLEN, &len, addr, &status);

// Change the AC source the 60 mV at 5MHz
send(addr, ":CVU:ACV 0.060", &status);
send(addr, ":CVU:FREQ 5E+6", &status);
```

```
// Get a new Z-measurement
send(addr, ":CVU:MEASZ", &status);
enter(recvstr, MAXLEN, &len, addr, &status);
```

Calling KULT user libraries remotely

KXCI contains a set of commands to call user libraries built by KULT on the Model 4200-SCS from a remote interface. Refer to Section 8 for details on using KULT.

There are five commands associated with the calling of user modules and are summarized in Table 9-13. Details on these commands follow the table.

Table 9-13
KXCI commands to call user modules

Command	Description
UL	Mode Switch: Switches KXCI to the usrlib mode.
EX	Execute: Executes the specified module in a KULT user library.
GN	Get Parameter By Name: Returns the parameter value (or values) for the specified input or output parameter. The parameter is specified by name.
GP	Get Parameter: Returns the parameter value (or values) for the specified input or output parameter. The parameter is specified by number. The first parameter after the command is number one.
GD	Get Parameter Description: Returns the description of the specified function. The function is specified by User Library and User Module.

UL: usrlib

The UL command is used to switch KXCI operation over to the usrlib mode. This command needs to be sent only once before any of the other commands to call user modules. To switch back to normal KXCI command modes, send DE or US.

Send the UL command through GPIB or Ethernet to change to the remote usrlib command set.

EX: execute

The EX command runs a user module using specified parameter values. The syntax for the command and parameters is shown as follows:

```
EX UserLibrary UserModule(param1, param2, param3....)
```

UserLibrary	The name of the User Library that contains the module to be run.
UserModule	The name of the User Module to be run.
param1, param2, param3, ...	The specified input or output parameters or both for the user module (function). See the programming notes for more information.

Programming notes:

- As shown in the syntax above, parameter values are to be separated by commas. Do not use quotation marks to enclose strings or names.
- **Input parameters** - For an input parameter, type in the value of the parameter. If the position for an input parameter is left empty, the default value for the parameter will be used.
- **Output parameters** - For an output parameter, leave the space empty (see example below).

Example 1

Assume that the following user module (built in KULT) performs a voltage sweep and stores the test voltages and measured current readings in arrays:

User Library: my_2nd_lib
User Model: VSweep

Also assume the parameter sequence for the VSweep function is as follows:

```
Vstart (input), Vstop (input), Imeas (output), NumIPoints (input),
Vforce (output), NumVPoints (input)
```

This example shows how that user function can be run from the KXCI console using parameters that perform a 11-point sweep starting at 0 V and stopping at 5 V.

With KXCI already in the usrlib mode, the following command will run the user's KULT function:

```
EX my_2nd_lib VSweep(0, 5, , 11, , 11)
```

After execution of the module completes, input or output parameters or both can be queried to return the values. Use GN: get parameter (by name) or GP: get parameter (by number) to query parameters.

GN: get parameter (by name)

The GN command is used to query input or output parameter values or both values by name for the last user module run in KXCI. The syntax for the command is shown as follows:

```
GN ParameterName NumValues
```

ParameterName	The name of the parameter in the KULT module.
NumValues	The number of values in an output array to be returned. See the programming notes for more information.

Programming notes:

- NumValues is only used for an output parameter that is an array.
- If NumValues is not used, one value will be returned.
- Arrays are returned as a list of values separated by semicolons.
- The value returned for an input parameter is the given value. The value(s) returned for an output parameter is the outcome of the test (for example, measured readings).

NOTE *A parameter can instead be queried by specifying the corresponding number for the parameter in the KULT module (see GP: get parameter (by number)).*

Example 2

The following command will query all 11 test voltages (Vforce parameter) for the function that was run in Example 1:

```
GN Vforce 11
```

The following array of test voltages will be returned and displayed in the KXCI console:

0; 0.5; 1.0; 1.5; 2.0; 2.5; 3.0; 3.5; 4.0; 4.5; 5.0

GP: get parameter (by number)

The GP command is used to query input or output parameter values or both values by number for the last user module run in KXCI. For example, Vforce in Example 1 is the fifth parameter.

The syntax for the command is shown as follows:

```
GP ParameterName NumValues
```

ParameterName	The name of the parameter in the KULT module.
NumValues	The number of values in an output array to be returned. See the programming notes for more information.

Programming notes:

- NumValues is only used for an output parameter that is an array.
- If NumValues is not used, one value will be returned.
- Arrays are returned as a list of values separated by semicolons.
- The value returned for an input parameter is the given value. The value(s) returned for an output parameter is the outcome of the test (for example, measured readings).

NOTE *A parameter can instead be queried by specifying the name of the parameter in the user module (see GN: get parameter (by name)).*

Example 3

The following command will query all 1 test voltages for the Vforce parameter for the user module that was run in Example 1. Vforce is the fifth parameter in the function:

```
GP 5 11
```

The following array of test voltages will be returned and displayed in the KXCI console:

0; 0.5; 1.0; 1.5; 2.0; 2.5; 3.0; 3.5; 4.0; 4.5; 5.0

GD – get description

The GD command is used to query and return the description for a KULT module. The syntax for the command is shown as follows:

```
GD User Library User Module
```

UserLibrary	The name of the User Library that contains the KULT module to be run
UserModule	The name of the User Module to be run

Example

The following command will query the description of the user module in Example 1. The description will be displayed in the KXCI console.

```
GD my_2nd_lib VSweep
```

SystemUtil User Library

This user library permits KXCI using the UL mode (see "Calling KULT user libraries remotely" on page 9-102) to retrieve information about the Model 4200-SCS instrument and the system.

NOTE *This user library is not compatible with KITE UTMs. It is designed to work with the UL mode of KXCI.*

Instrumentinfo Retrieve Model 4200-SCS instrument card information.

Purpose Retrieve all information on the instrument cards in the Model 4200-SCS.

Format

```
int instrumentinfo(char *result, int maxlen, int *len)
```

*result: String of results for the instruments.

maxlen: The maximum number of bytes that the result can stored in the buffer.

**len*: Number of bytes returned by the function

Remarks This function returns all of the instrument-level information for the cards in the Model 4200-SCS. The results contains the following information for each instrument card in the Model 4200-SCS chassis: slot number, instrument card ID, model number, serial number, hardware version, firmware version, calibration date, calibration due date. All of the information is comma-separated, see Example below.

Example Output for a Model 4200-SCS with 3 instrument cards installed in slot 3 (Model 4220-PGU), slot 5 (Model 4225-PMU), slot 7 (Model 4210-CVU):

```
slotno:3,name:VPU1,model:KIVPU4220,serialno:1254281,hwver
:1.0,fwver:1.50,caldate:Dec 19, 2009,caldue:Dec 19, 2010,
slotno:5,name:PMU1,model:KIPMU4225,serialno:1276563,hwver
:1.0:544911,fwver:1.50,caldate:Nov 28, 2011,caldue:Nov
27, 2012,
slotno:7,name:CVU1,model:KICVU4210,serialno:Z005712,hwver
:3.0:493083,fwver:2.03,caldate:Aug 09, 2010,caldue:Aug
09, 2011
```

Systeminfo Retrieve Model 4200-SCS system information.

Purpose Retrieve system level information for the Model 4200-SCS

Format

```
int systeminfo(char *      , int       , int *   );
```

* : String of results for the system.

: The maximum number of bytes that the result can stored in the buffer.

* : Number of bytes returned by the function

Remarks This function returns system level information for the Model 4200-SCS. The results string contains the following information: Model 4200-SCS serial number, system software version, system platform version, KTEI software version. The results are comma-separated, see the Example below.

Example System info output for a Model 4200-SCS, running KTEI 8.2 Service Pack 3:

```
serialno:1209478,swver:4200-852-6.0,platformver:4200-
300-3B,kteiver:V8.2-SP3
```

KXCI Ethernet client driver

A driver (single DLL) is provided to control KXCI through the Ethernet. This driver can be copied to the user's controlling computer. The DLL is standalone (atomic). That is, it does not depend on any other DLLs, so it can be easily moved/copied.

This driver DLL is named KXCIClient.DLLand is located at the following command path:

```
C:\S4200\sys\bin
```

The KXCIClient.lib is located at the following command path:

```
C:\S4200\sys\lib
```

For convenience, a C header file (KXCIClient.h) is included and has the above prototypes. It is located at the following command path:

```
C:\S4200\sys\include
```

Driver functions

The KXCIClient.DLL driver has the following functions:

int OpenKXCIConnection_C(char **IPAddrStr*, int *PortNum*, int **err*);

IPAddrStr	IP address in sting format "nnn.nn.nn.nn" (that is, 129.22.35.17).
PortNum	IP Port assigned in KXCI tab of KCON.
err	Socket err returned by WSAGetLastError().

int SendKXCICommand_C(char **cmdstr*, char **ReturnString*, int **err*);

cmdstr	KXCI command string, that is, "DE;CH1;CH2".
ReturnString	Data returned by command, if any. If input command results in data to be returned, it is place here.
err	Socket err returned by WSAGetLastError().

int GetKXCISpollByte_C(unsigned short **spbyte*, int **err*);

spbyte	KXCI spoll byte (same as GPIB byte).
err	Socket err returned by WSAGetLastError().

void CloseKXCIConnection_C(void);

Section 10
System Administration

In this section:

Introduction

Because the Keithley Instruments Model 4200-SCS Semiconductor Characterization System contains an embedded PC and is a networkable instrument, additional information is often needed to efficiently administer the system in a multi-user or multi-system environment. Section 10 discusses the following relevant topics:

- **Embedded PC policy**: Describes the policy that must be adhered to when modifying the software installed on the Model 4200-SCS.
- **Default user accounts**: Describes the properties of the preconfigured user accounts and how to log onto them.
- **Managing multiple users and Model systems**: Discusses management techniques for multi-user sharing of single and multiple Model 4200-SCS systems.
- **Disk defragmenter**: Briefly describes how to run the preinstalled disk defragmentation software.
- **Default BIOS and video settings:** Provides a summary of the default BIOS and video settings.
- **System-level backup and restore software**: This non-Keithley software tool allows Model 4200-SCS users to perform software backups, as well as restore the Model 4200-SCS to factory condition.
- **Placing KITE or KXCI in the Windows Startup menu**: A procedure for Microsoft® Windows® operating system.

NOTE *Most of the procedures discussed in this section should be performed by a knowledgeable Microsoft® Windows® system administrator.*

Embedded PC policy

CAUTION **Keithley Instruments warrants the performance of the Model 4200-SCS only with the factory-approved Windows Operating System and application software preinstalled on the Model 4200-SCS by Keithley Instruments. Systems that have been modified by the addition of unapproved non-Keithley application software (software that is not explicitly approved and supported by Keithley Instruments) are not covered under the product warranty. Model 4200-SCS systems with unapproved software may need to be restored to factory-approved condition before any warranty service can be performed (for example, calibration, upgrade, technical support). Services provided by Keithley Instruments to restore systems to factory-approved condition will be treated as out-of-warranty service with associated time and material charges.**

CAUTION **DO NOT reinstall or upgrade the Windows operating system (OS) on any Model 4200-SCS. This action should only be performed at an authorized Keithley Instruments service facility. Violation of this precaution will void the Model 4200-SCS warranty and may render the Model 4200-SCS unusable. Any attempt to reinstall or upgrade the Windows Operating System (other than a Windows service pack update) will require a return-to-factory repair and will be treated as an out-of-warranty service, including time and material charges.**

Although users must not attempt to reinstall or upgrade the operating system, the user can restore the hard drive image (complete with OS) using the Acronis True Image OEM, described in Section 10 "System-level backup and restore software."

Default user accounts

Preconfigured user accounts

There are three preconfigured Windows user accounts on the Model 4200-SCS, each of which is described below.

The "kiuser" account

By default, the **"kiuser"** account logon should be used by all Model 4200-SCS users. Multiple users can share this logon account and still store all of their data in individual user directories. This account allows access to all of the KTE Interactive software tools and applicable non-Keithley software tools and offers the same access to Windows and KTE Interactive software tools as the "kiadmin" account.

Logon name: **"kiuser"**

Password: None is required. Press **ENTER**.

The "kiadmin" account

The **"kiadmin"** account logon should be used by Model 4200-SCS system administrators only. This account provides access to **Start > Programs > Administrative Tools**, as well as to all of the KTE Interactive software tools.

Logon name: **kiadmin"**

Password: **"kiadmin1"**

Creating new user accounts

Because Windows is a multi-user operating system, it is possible to create unique Windows® user accounts for each Model 4200-SCS user. This enables you to customize the behavior of a Model 4200-SCS without affecting its behavior for other users. Similarly, it provides additional data protection and privacy, because each user can log onto the Model 4200-SCS using a unique logon name and password.

CAUTION **Setting up multiple Windows® user accounts is an advanced system administration procedure that should be performed only by a knowledgeable Windows system administrator.**

To set up a new user account, follow these two basic steps:

1. Perform the following procedure:
   - Windows: Select **Start > Control Panel > User Accounts** and then **Create a new account**.
2. Log on as **<newuser>** and set up the desktop and other elements of the **<newuser>** profile. In addition, run the KTE Interactive **Initialize New User** utility (**Start > Programs > Keithley > Initialize New User**) to properly configure KTE Interactive for the new user.
3. Log on as **<newuser>** and set up the desktop and other elements of the **<newuser>** profile. In addition, run the KTE Interactive **Initialize New User** utility (**Start > Programs > Keithley > Initialize New User**) to properly configure KTE Interactive for the new user.

NOTE ***Initialize New User*** *must be run each time a new user account is created; it does not require user input and only needs to be run once per user account.*

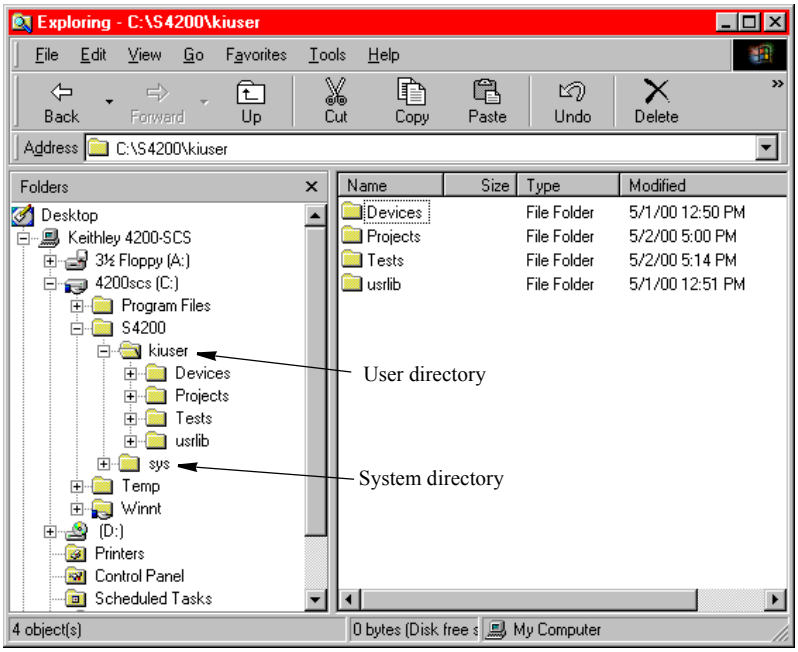
Managing multiple users and systems

When multiple users share one or more Model 4200-SCS systems, the following actions are often necessary or desirable:

- To effectively manage data and tests from multiple users, use personal user directories.
- To share data between users and/or between Model 4200-SCS systems, store data in a network file system.
- To store user data on a network, using a network file system.

By default, all KTE Interactive files are stored in the `C:\S4200` folder, as illustrated in Figure 10-1 below:

Figure 10-1
KTE Interactive file structure



Most of the KTE Interactive files can be stored on any disk drive that is accessible by the Model 4200-SCS. The following subsections further define the underlying file structure, and discuss the process of configuring the KTE Interactive software tools to utilize other storage locations. Additional information regarding KTE Interactive file management can be found in an expanded version of Default user directory: C:\S4200\kiuser in Section 6.

NOTE *Windows networking software is preinstalled on the Model 4200-SCS, along with a network interface card (NIC) driver. In most cases, basic TCP/IP configuration is all that is required to be able to share data between Model 4200-SCS systems using My Network Places. Refer to the Windows documentation provided with the Model 4200-SCS for additional information.*

Default user directory C:\S4200\kiuser

By default, all of the sample projects and standard libraries that are included with each version of KTE Interactive are stored in this directory. However, projects and libraries can be stored (and shared) on any accessible disk drive, including a network drive. The subdirectories in the default user folder contain the following information:

- **Devices**: Default KITE device library that is provided with each version of KTE Interactive. By default, all Model 4200-SCS users can access this device library when using KITE.
- **Projects**: Default KITE project library that is provided with each version of KTE Interactive. By default, all Model 4200-SCS users store projects in this directory.
- **Tests**: Default KITE test library that is provided with each version of KTE Interactive. By default, all Model 4200-SCS users can access this test library when using KITE.
- **usrlib**: Default collection of KULT user libraries that is provided with each version of KTE Interactive. By default, all Model 4200-SCS users have access to these user libraries when using KULT and KITE.

System directory C:\S4200\sys

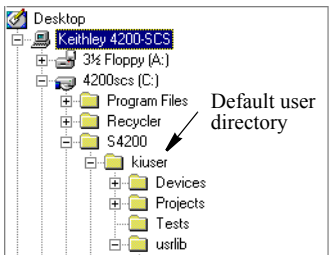
The system directory stores all of the binary and executable files that KTE Interactive needs to control the Model 4200-SCS.

CAUTION **The files stored in the `C:\S4200\sys` folder must not be modified in any way, neither by Model 4200-SCS users, *nor by system administrators*. This folder (directory) must reside on the Model 4200-SCS hard disk.**

Creating additional user or personal directories

When the KTE Interactive software is installed, all user files are stored in the `C:\S4200\kiuser\` directory. Figure 10-2 highlights this directory.

Figure 10-2
The "default" C:\S4200\kiuser\ directory



By default, the `C:\S4200\kiuser\` directory is the only active user directory. However, the ability to work with *alternative* user directories is often desirable, as in the following two situations:

- Multiple users share a Model 4200-SCS. It is desirable that each user works with a personal directory, which is stored in a separate location.
- Multiple Model 4200-SCS instruments are installed on a local area network (LAN). It is desirable for all users to be able to access a single KULT user library that is stored on the server of the LAN.

The KTE Interactive software allows you to add and work with libraries in alternative directories.

Adding personal user directories

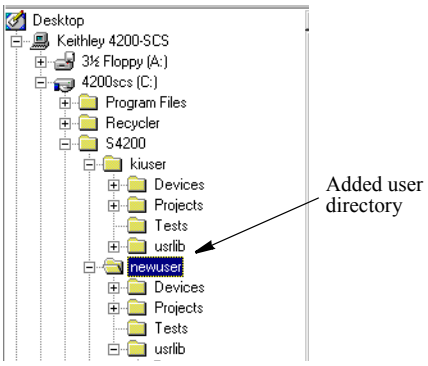
When the Model 4200-SCS is used in a multi-user environment, all libraries and projects may be stored in unique locations for each user. To set up these individual locations most easily, do the following:

1. Create a new user directory under `C:\S4200`, for example: `C:\S4200\newuser`.

NOTE *To avoid confusion amongst Model 4200-SCS users, give the directory the same name as the user.*

2. Copy the contents of the `C:\S4200\kiuser` directory to the Windows clipboard.
3. Paste the contents of the clipboard into the new directory. Figure 10-3 shows a directory created for an individual user called `newuser`.

Figure 10-3
Added personal user directory



4. Before using the KULT user libraries that are present in the new user directory (`C:\S4200\newuser\usrlib`), update these libraries as follows:
   a. Set %KI_KULT_PATH%[1] to `C:\S4200\newuser\usrlib` (refer to Adding a directory that contains network shared user libraries in Section 8).
   b. Run the `kultupdate` command-line utility. Enter the following commands at the command line:

```
C:\>kultupdate Winulib
C:\>kultupdate matrixulib
C:\>kultupdate ki590ulib -dep Winulib
C:\>kultupdate ki42xxulib
C:\>kultupdate hp8110ulib
C:\>kultupdate hp4284ulib
```

   For details about the `kultupdate` utility, refer to Updating user libraries using kultupdate in Section 8.
5. Repeat steps 1 through 4 for each new Model 4200-SCS user.

CAUTION **After creating personal KULT user libraries (for example, the libraries in `C:\S4200\newuser\usrlib`), users must ensure that they access and work with their own user libraries. This precaution avoids potential errors caused by connecting one user's UTMs to another user's user modules or, worse, editing another user's user libraries. Consequently, *each time* before using the Model 4200-SCS, *the user must run KCON and set the* `%KI_KULT_PATH%` variable to the intended user library directory. Refer to "Changing the active user-library directory" in Section 8.**

Adding a directory that contains network shared user libraries

User libraries can be stored on a local area network (LAN) so that they can be shared. To set this up, refer to Adding a directory that contains network shared user libraries in Section 8.

1. `%KI_KULT_PATH%` specifically, and `%NAME%` generally, are environment variables. Each such environment variable is a string variable that stores a directory-path string. For example, the content of `%KI_KULT_PATH%` is the location where KITE and KULT look for user libraries and user modules. The default content of `%KI_KULT_PATH%` is `C:\S4200\kiuser\usrlib`. Use KCON or the `set` command-line utility to change the content of `%KI_KULT_PATH%` to another location, for example, to a personal user-library location, such as `C:\S4200\Your-Name\usrlib`. For more information about changing the content `%KI_KULT_PATH%`, refer to "Changing the active user-library directory" in Section 8.

Sharing libraries and projects

By default, KITE users can select tests only from `C:\S4200\kiuser\tests`, and can select devices only from `C:\S4200\kiuser\devices`. However, it is possible to share tests and devices from other device and test library directories by adding them to the KITE selection lists (for example, as found in KITE Subsite Plan and Device Plan windows).

By default, KITE users select projects from and submit projects to `C:\S4200\kiuser\Projects`. However, projects may likewise be shared by saving them to alternative project directories, including network project directories.

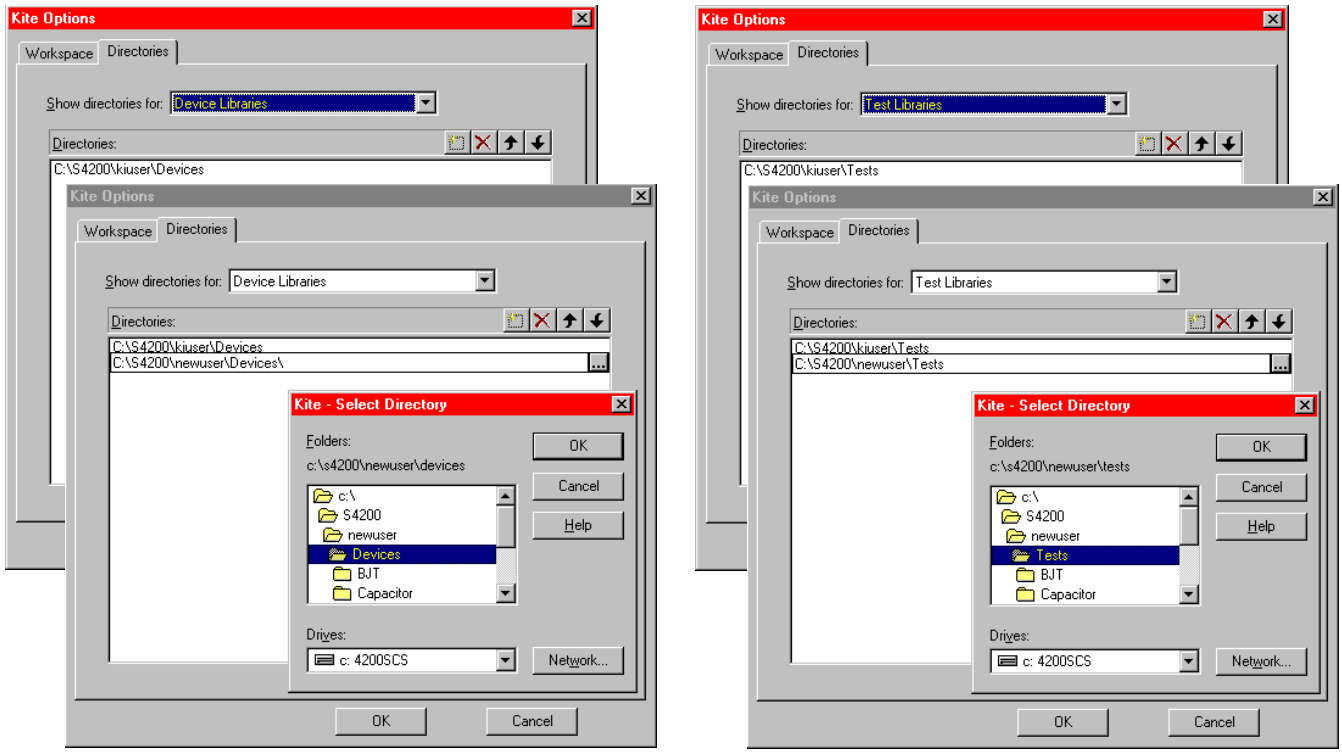
Finally, it is possible to share user libraries from alternative directories, but only one at a time. KULT and KITE can access only one user library directory in a given session. By default, this active library is `C:\S4200\kiuser\usrlib`. However, the Keithley CONfiguration (KCON) program allows you to set an alternative user library directory to be the active user library directory.

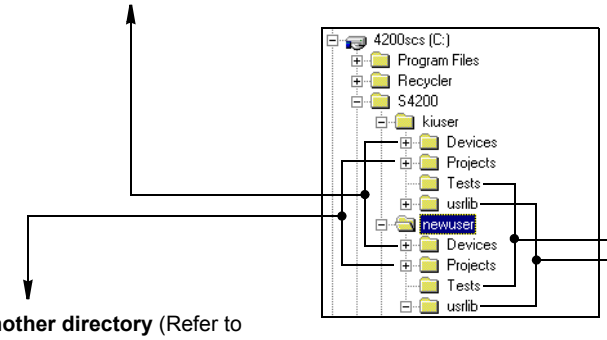
Four groups of illustrations in Figure 10-4 summarize how to: 1) share test and device libraries; 2) save projects to alternative directories; and 3) specify the active user library for KITE and KULT. Each summary references a more detailed procedure that is located elsewhere in the Model 4200-SCS Reference Manual.

Figure 10-4
Project/library sharing summary

Adding a device directory to KITE (Refer to "Specifying which device library directories are to be available to projects" in Section 6.)

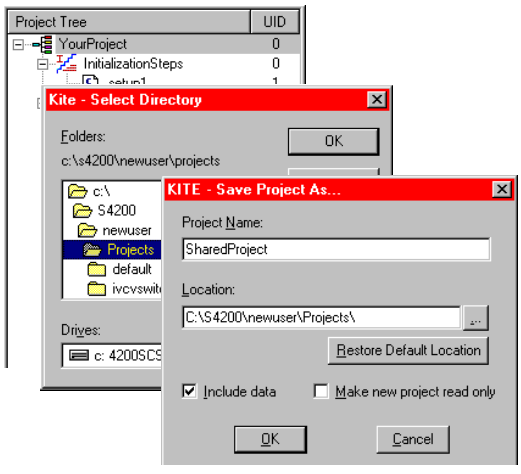
Adding a test directory to KITE (Refer to "Changing the active user-library directory" in Section 8.)

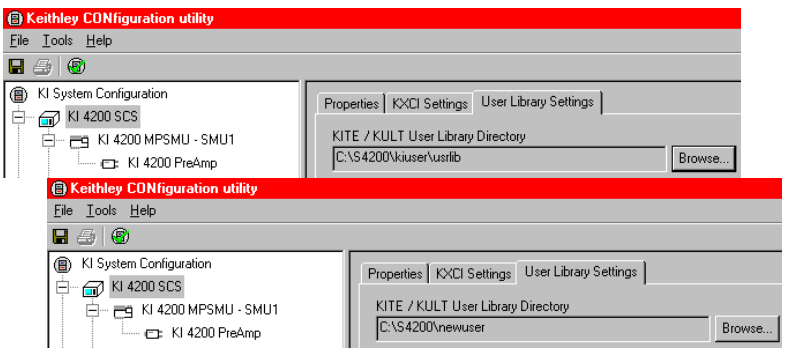




Saving a project to another directory (Refer to "Saving a Project Plan under a new name" in Section 6.)

Changing the KITE/KULT user-library directory (Refer to "Specifying which test library directories are to be available to projects" in Section 6.)





Default BIOS settings

As with any PC, the embedded PC inside the Model 4200-SCS uses a programmable BIOS device to store basic I/O and hardware-configuration software. This software is preconfigured at the factory to optimize the performance of the Model 4200-SCS, and, the configuration typically will not need to be changed. The preconfigured BIOS settings for the Model 4200-SCS are shown in Figure 10-5 through Figure 10-19.

NOTE *To access the **configure BIOS** configuration settings, hold down the **DEL** key while system is powering up.*

The screenshots below illustrate the BIOS for the 4200 System Platform Versions: 4200-300-3A, 4200-300-3B, 4200-300-3C, or 4200-300-3D. The platform version can be found in KCON (see Figure 7-1 on page 7-4).

Figure 10-5
Standard CMOS Features screen

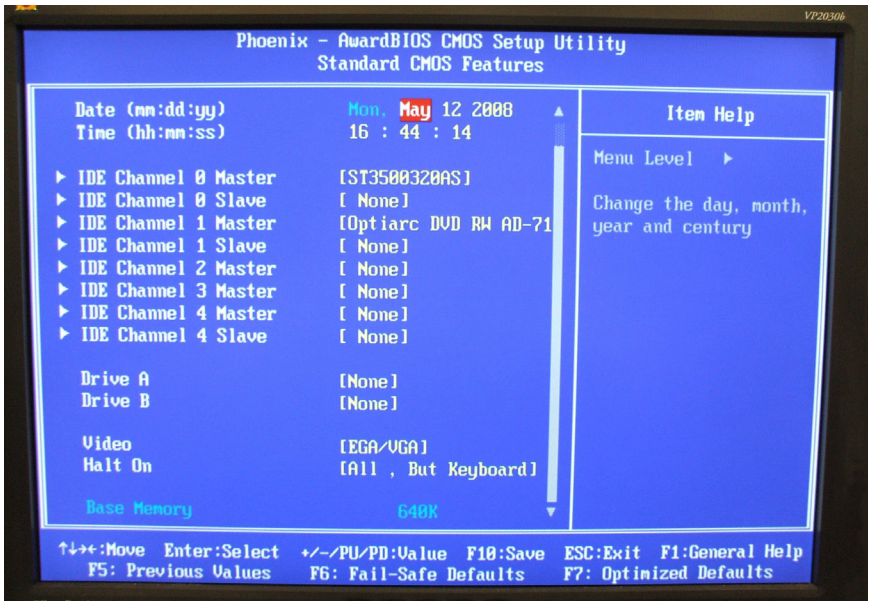


Figure 10-6
Advanced BIOS Features screen (Part 1)

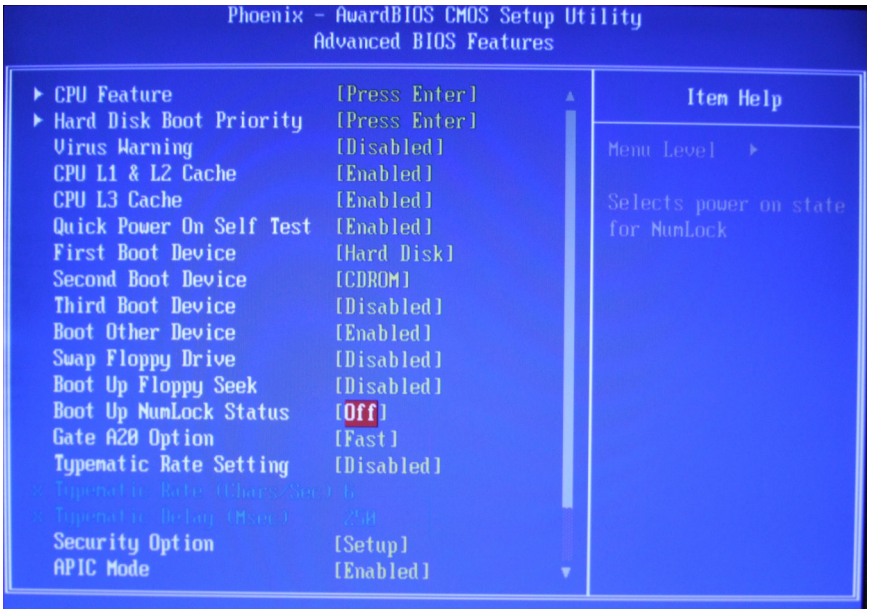


Figure 10-7
Advanced BIOS Features screen (Part 2)

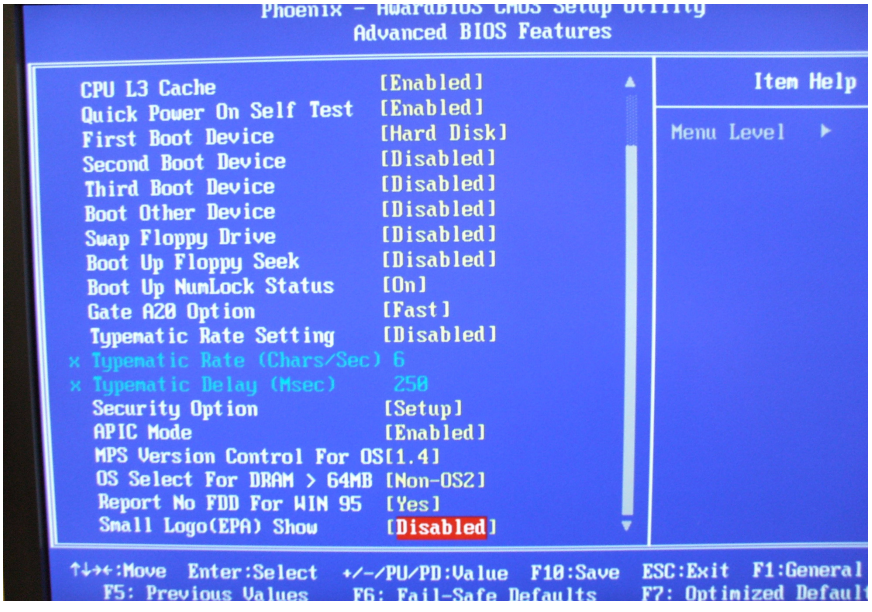


Figure 10-8
CPU Features screen

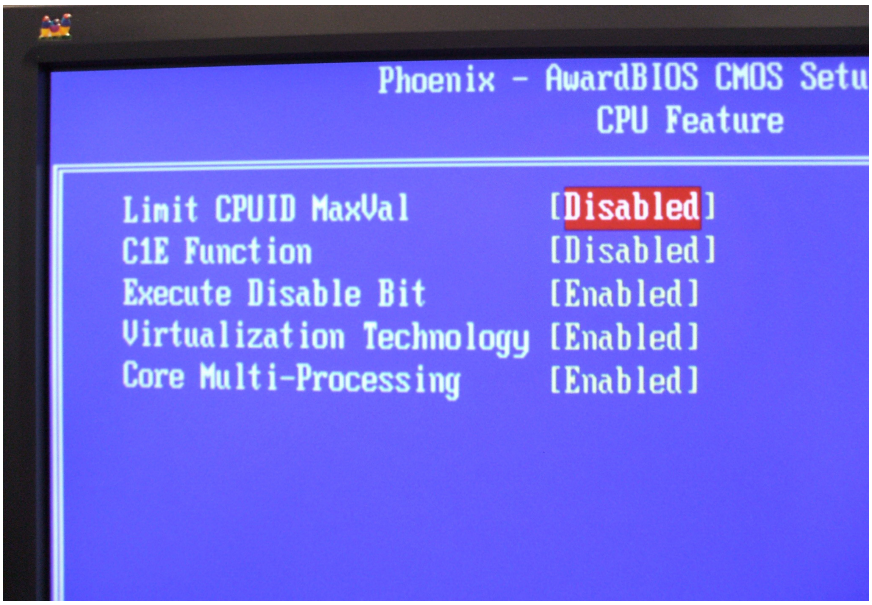


Figure 10-9
Hard Disk Boot Priority screen

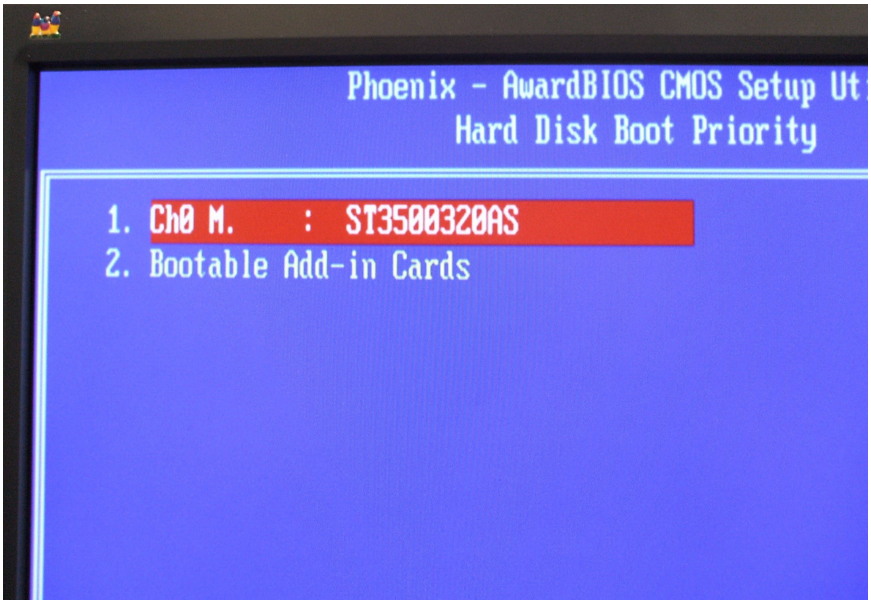


Figure 10-10
Advanced Chipset Features screen

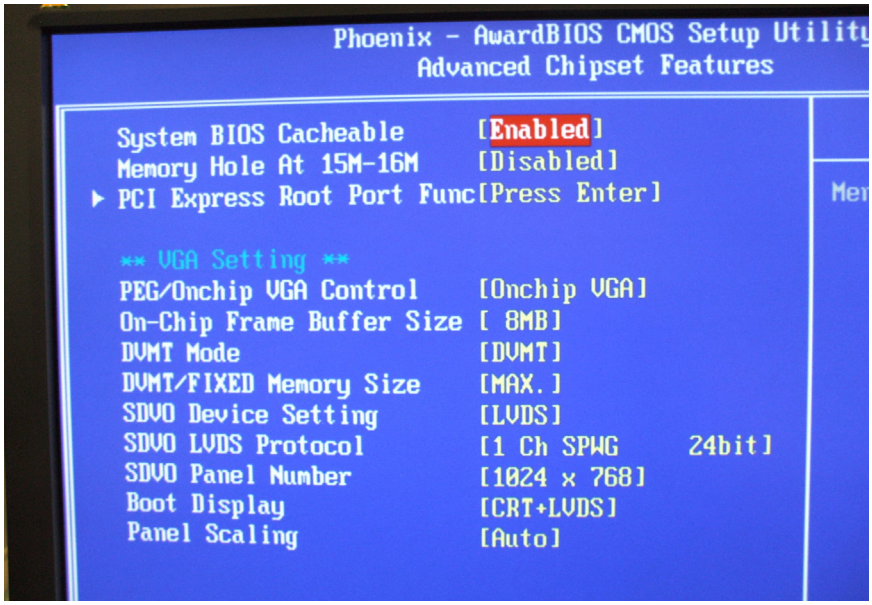


Figure 10-11
PCI Express Root Port Function screen

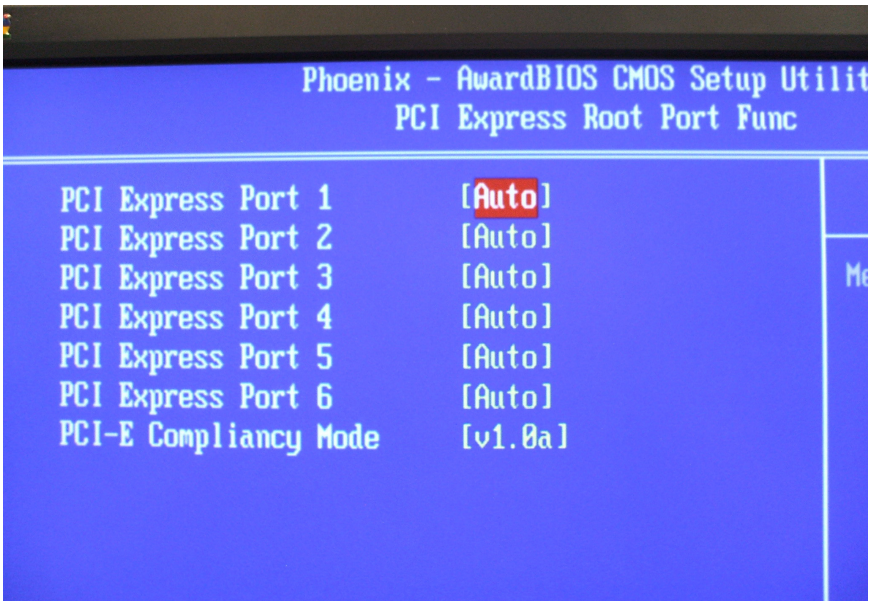


Figure 10-12
OnChip IDE Device screen

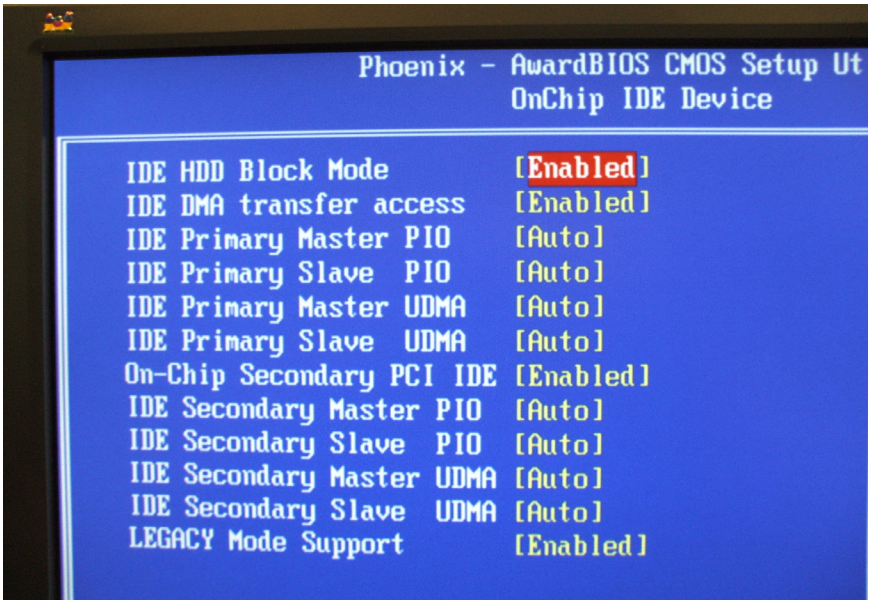


Figure 10-13
Super IO Device screen

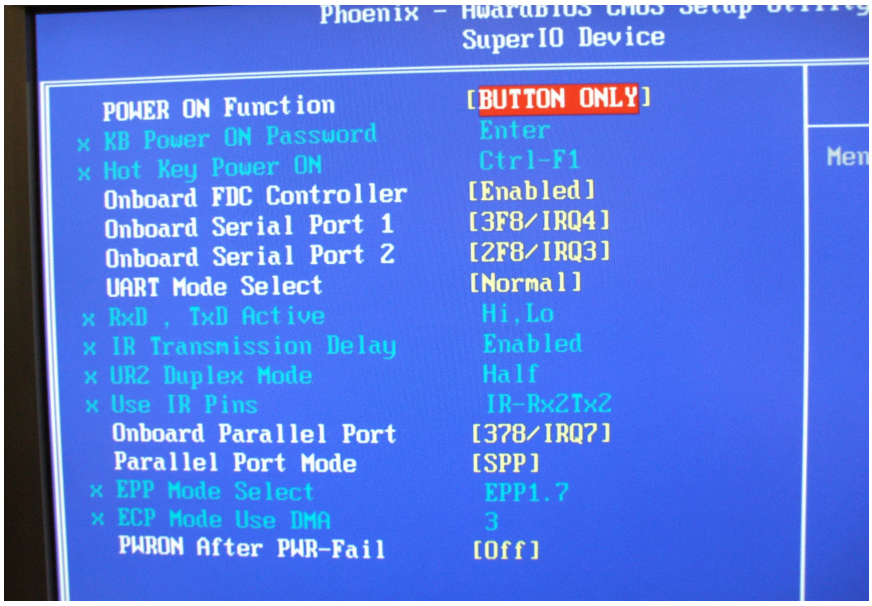


Figure 10-14
USB Device Setting screen

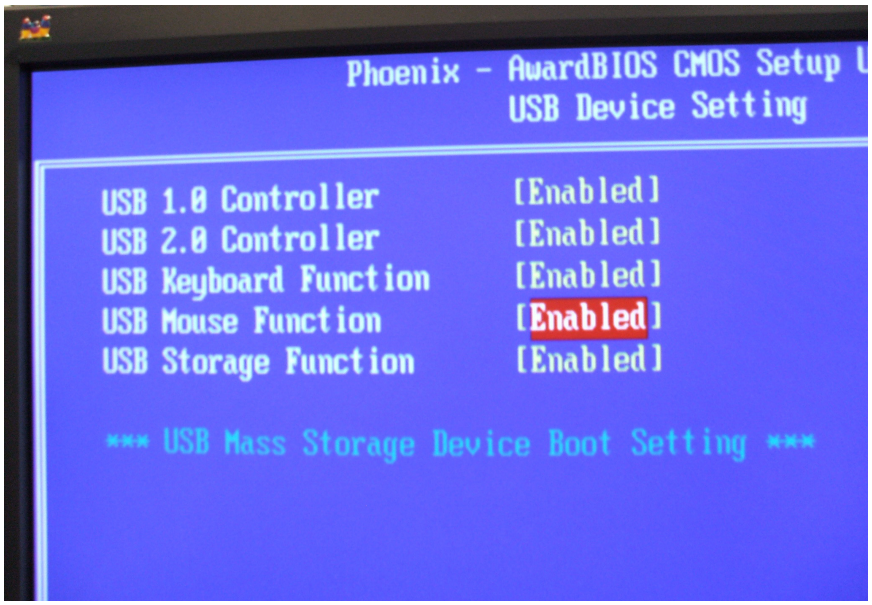


Figure 10-15
Power Management Setup screen (Part 1)

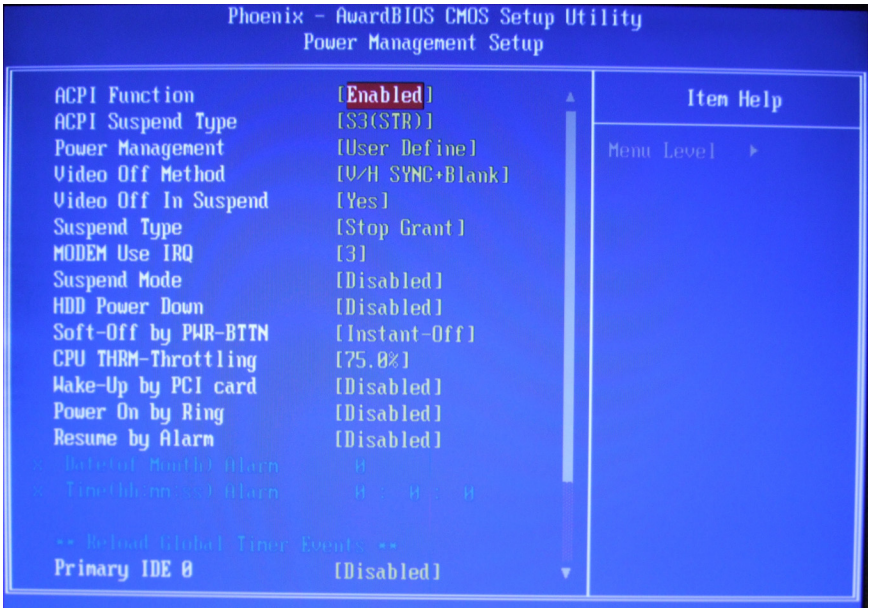


Figure 10-16
Power Management Setup screen (Part 2)

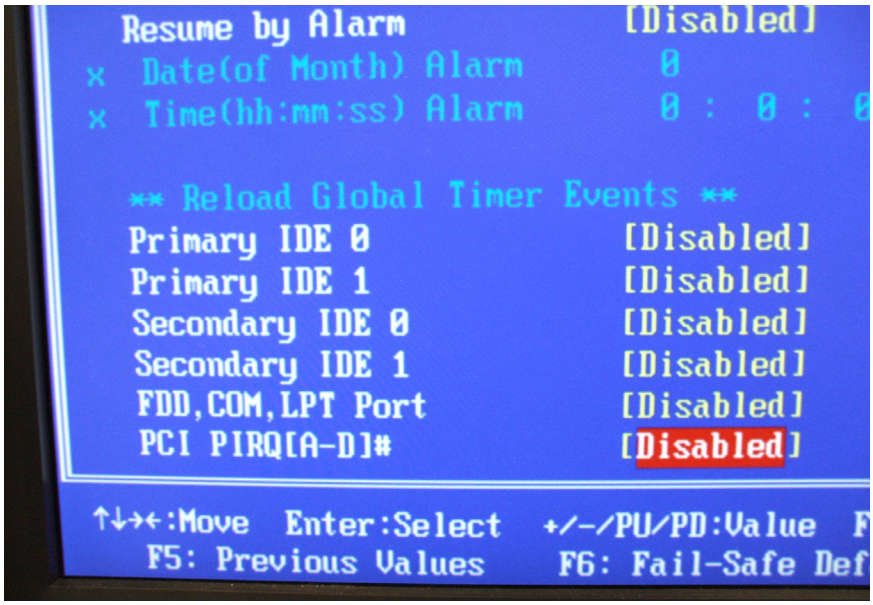


Figure 10-17
PnP/PCI Configurations screen

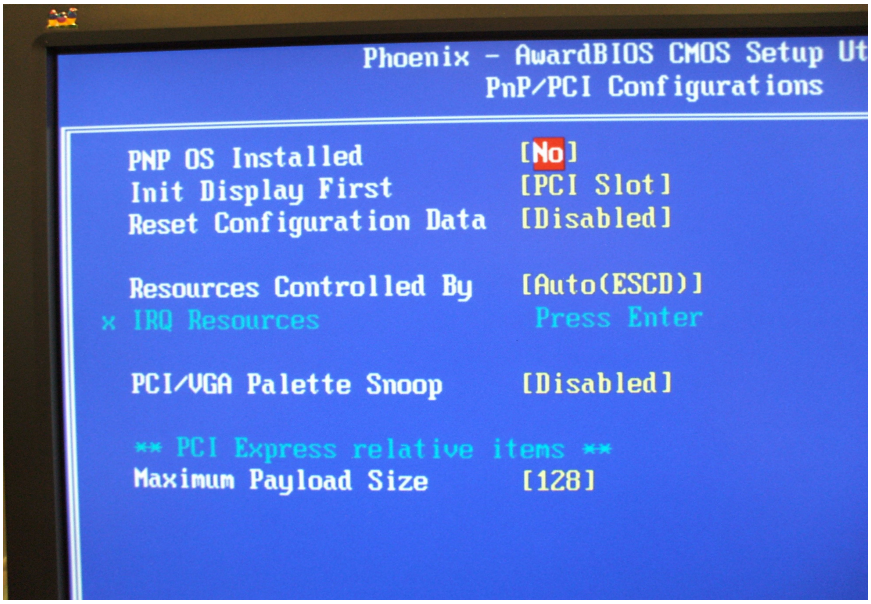


Figure 10-18
PC Health Status screen

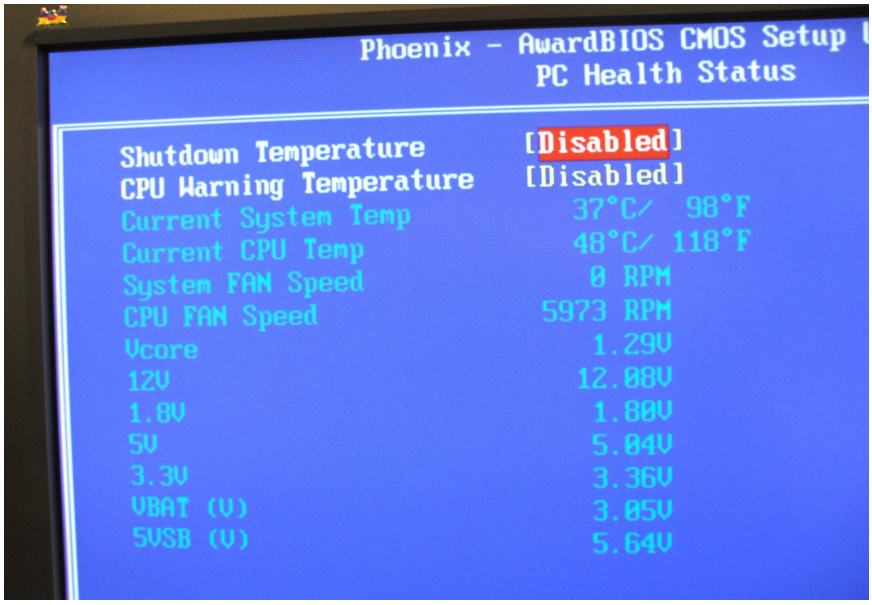
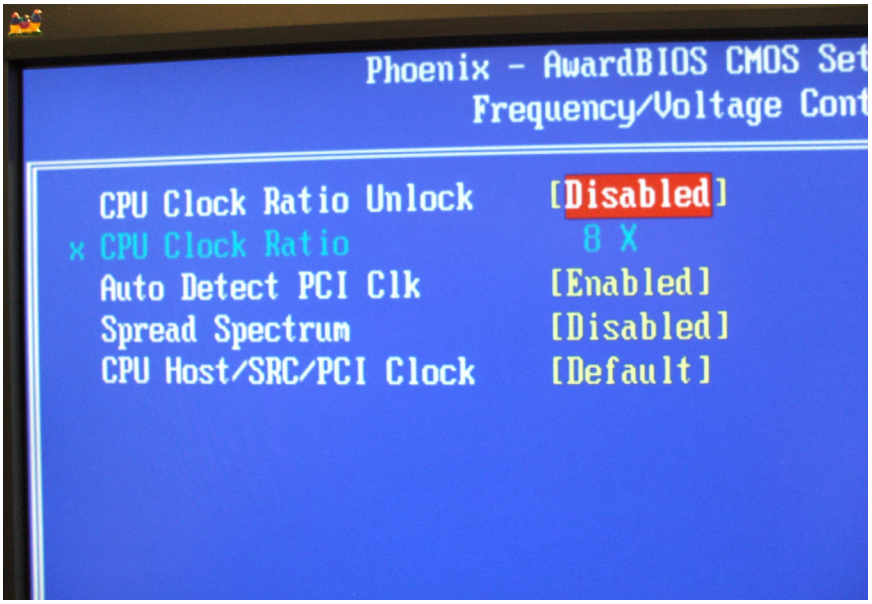


Figure 10-19
Frequency/Voltage Control screen



Default video settings

The Model 4200-SCS can be ordered with or without an integrated flat panel display (FPD). A system with integrated FPDs (Model 4200-SCS/F) has some unique capabilities and attributes, that are discussed below.

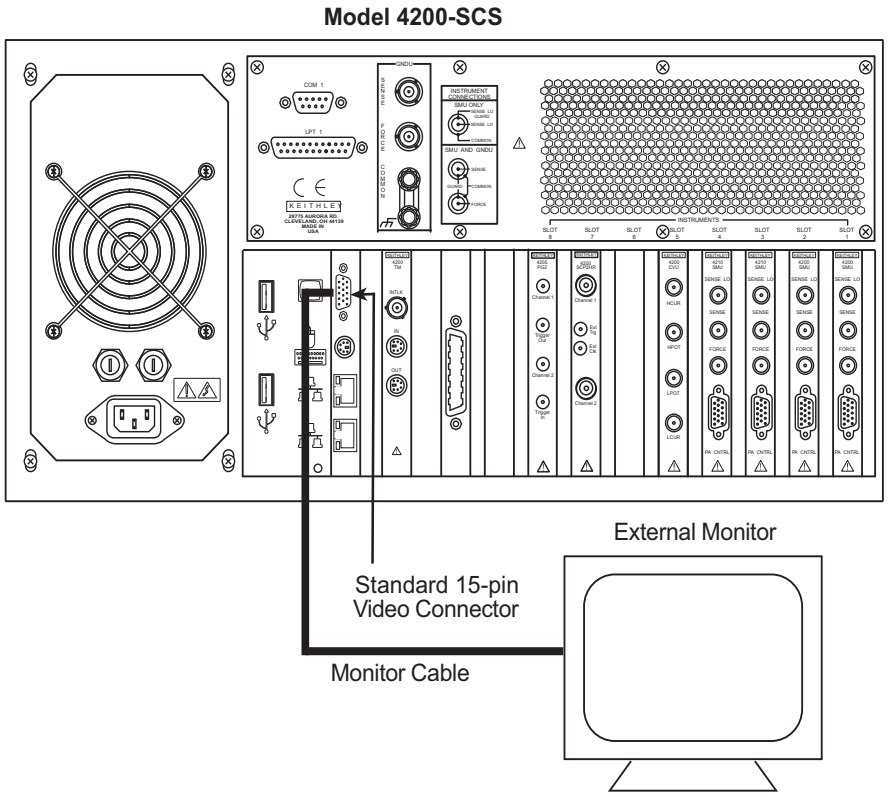
High resolution flat panel display

All Model 4200-SCS systems shipped after December 2005 include a high-resolution (1024x768) flat panel display. The high-resolution display is also available for older machines through an upgrade.

Driving an external monitor from a 4200-SCS with an integrated FPD

A Model 4200-SCS with an integrated FPD can drive an external monitor, or both the external monitor and the integrated FPD simultaneously. See Figure 10-20.

Figure 10-20
External monitor connections



NOTE *All 2GHz and Core 2 Duo Model 4200-SCS FPD systems are shipped with the external connector active.*

Placing KITE or KXCI in the Windows startup menu

KITE or KXCI can be placed in the Windows® **Startup** folder. When the appropriate user(s) logs into Windows, KITE or KXCI will start automatically.

1. Open Windows Explorer by clicking the Windows Explorer icon on the Quick Launch Bar at the bottom of the Desktop. Make sure the KITE or KXCI icon on the Desktop is visible. If not, reduce the size of Windows Explorer such that the Desktop icons can be viewed.
2. Use Windows Explorer to open the **Startup** folder for **All Users**:
   Local Disk (C:)
   Documents and Settings
   All Users

Start Menu
Programs
Startup

3. Right-click on the KITE or KXCI icon and drag it to the **Startup** folder in Windows Explorer.
4. Release the mouse button and select **Copy Here** from the pop-up menu. The next time Windows is started, KITE or KXCI will start automatically.

NOTE *The above procedure configures automatic KITE or KXCI startup for all users. You can instead configure automatic startup for a specific user (**kiuser** or **kiadmin**). For a specific user, repeat the above procedure, but instead of opening the **All Users** folder, open the **kiuser** or **kiadmin** folder.*

*If you configure automatic KITE or KXCI startup for a specific user, you must remove KITE or KXCI from the **Startup** folder for **All Users** if it was added. Otherwise, system conflicts will occur (see following **CAUTION**).*

CAUTION **Add a shortcut for only one of these two programs: KITE OR KXCI. Only one of these programs may run at a given time. Consequently, adding Startup shortcuts for both programs results in system conflicts.**

System-level backup and restore software

NOTE *All new Model 4200-SCS systems shipped after December 2005 are shipped with a new non-Keithley software tool that will provide users with system-level backup and restore capability.*

Acronis True Image OEM

Acronis True Image (OEM) is a software tool that allows Model 4200-SCS users to create hard-disk images including user data, environment settings, and operating system files. This software is preinstalled on every Model 4200-SCS system at the Keithley Instruments factory.

Please refer to your Acronis True Image OEM Edition User Guide for further details. This user guide can be found by opening the Model 4200 Complete Reference and clicking on Related Literature (Figure 10-21), then selecting the Acronis True Image OE User Guide (Figure 10-22).

Figure 10-21
Model 4200-SCS Complete Reference screen (below) to access the desired information.

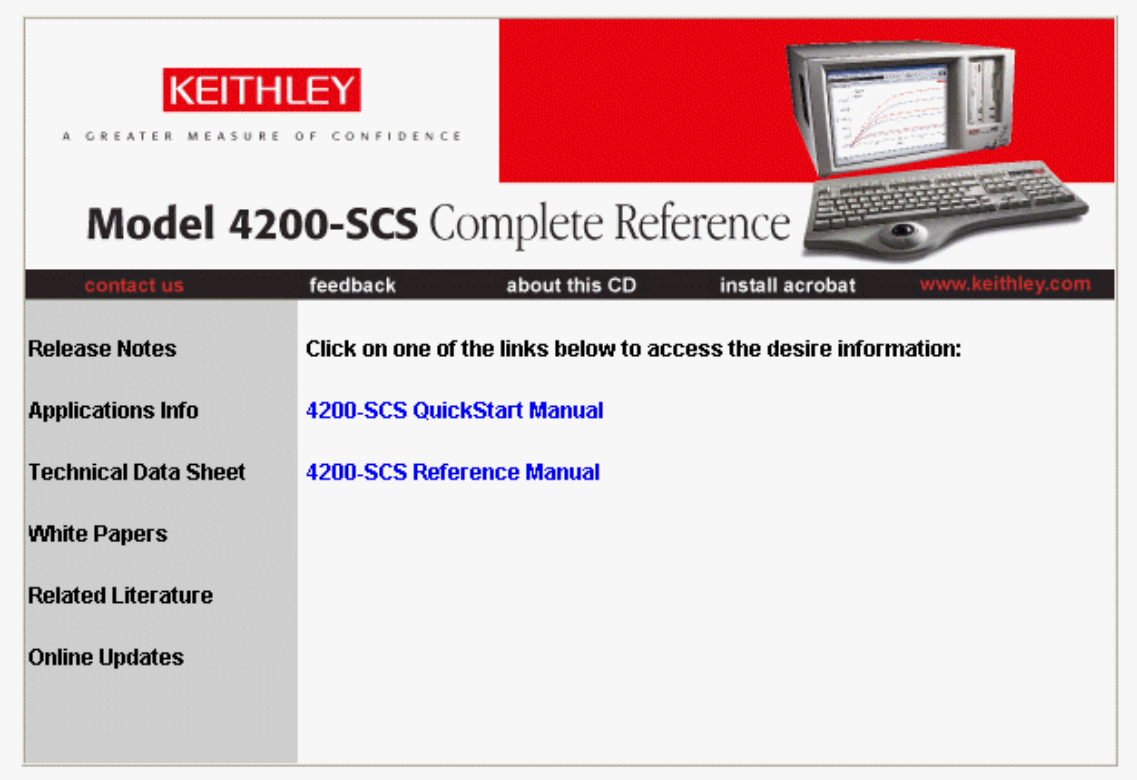


Figure 10-22
Model 4200-SCS Related Literature screen

Image restore to factory condition

In addition to allowing users to create their own backups to the hard drive, CD-ROM, flash drive, or other media, the entire Model 4200-SCS hard drive image has been archived just prior to shipment. This will allow the user at any time to restore the entire contents of the Model 4200-SCS hard drive to the exact condition it was in just prior to shipment from the Keithley Instruments factory to the customer.

To restore the Model 4200-SCS hard drive to factory condition, simply reboot the Model 4200-SCS. During the BIOS bootup process, the following message will display for approximately three seconds:

```
Press <F11> for Acronis Startup Recovery Manager
```

Pressing <F11> at this time will start an Acronis Bootup program that will allow the Model 4200-SCS factory hard drive image to be restored. Simply follow the instructions in the boot manager. The entire restore process will take approximately 10 minutes.

CAUTION **Restoring the Model 4200-SCS hard drive to factory condition will DELETE ALL files written to the hard drive after shipment. Please ensure you back up all data and files PRIOR to restoring the hard drive to factory condition.**

Section 11

Pulse Source-Measure Concepts

In this section:

NOTE *Basic Pulse hardware and concepts are covered in the Model 4200-SCS User's Manual, Pulse cards, .*

Settings for pulse generator

Table 11-1
Setting for pulse generator

Setting or feature	Description	Access level	Default	LPT function	Pulse mode		
					Standard	Full Arb	Seg-Arb
Sets pulse output mode	Swaps the values for pulse high and pulse low	Channel	Normal Pulse	pulse_output_mode	√	(n/a)	(n/a)
Current limit	Sets the current limit for pulse output	Channel	105 mA	pulse_current_limit	√	√	√
DC Mode	Selects fixed DC voltage output and sets the level	Channel	(n/a)	pulse_dc_output	√	(n/a)	(n/a)
Full Arb waveform	Defines a Full Arb waveform	Channel	(n/a)	arb_array	(n/a)	√	(n/a)
Full Arb waveform file	Loads a waveform from a Full-Arb waveform file	Channel	(n/a)	arb_file	(n/a)	√	(n/a)
High Endurance Output Relay	Controls the high endurance solid-state output relay	Channel	Closed, Auto	pulse_ssrc	√	√	√
Pulse count	Sets the number of pulses for burst output.	Channel	1	pulse_burst_count	√	√	√
Pulse delay	Sets the time delay from trigger to pulse output.	Channel	0 s	pulse_delay	√	√	√
Pulse halt	Stops all pulse output	Card	(n/a)	pulse_halt	√	√	√
Pulse high/low	Sets the pulse high and low level	Channel	0 V	pulse_vhigh and pulse_vlow	√	(n/a)	(n/a)
Pulse load	Sets the impedance of the attached device under test	Channel	50Ω	pulse_load	√	√	√
Pulse output	Turns pulse output on or off	Channel	Off	pulse_output	√	√	√
Pulse period	Sets the period for pulse output	Card	1µs	pulse_period	√	(n/a)	(n/a)
Pulse trigger output	Enables or disables output triggers	Card	On	pulse_trig_output	√	√	√
Pulse trigger source	Sets the input trigger source	Card	Software	pulse_trig_source	√	√	√
Pulse Width	Sets the pulse width	Channel	500 ns	pulse_width	√	(n/a)	(n/a)
Reset	Returns pulse card to default settings*	Card	(n/a)	pulse_init and pg2_init	√	√	√
Rise/fall time	Sets the pulse rise/fall time	Channel	100 ns	pulse_rise and pulse_fall	√	(n/a)	(n/a)
Segment ARB waveform**	Defines a Segment ARB® waveform	Channel	(n/a)	seg_arb_define	(n/a)	(n/a)	√
Segment ARB waveform file**	Loads a waveform from a Segment-Arb waveform file	Channel	(n/a)	seg_arb_file	(n/a)	(n/a)	√
Source range	Sets the voltage range for pulse output	Channel	5 V; high speed	pulse_range	√	√	√
Trigger mode	Triggers start of Continuous, Burst or Trig Burst	Card	Continuous	pulse_trig	√	√	√
Trigger polarity	Sets the polarity for Trigger Output	Card	Positive	pulse_trig_polarity	√	√	√

* There are two functions to reset the pulse generator card:
pulse_init – Resets the unit to the defaults for the presently selected pulse mode (Standard, Full Arb or Segment ARB).
pg2_init – Selects the specified pulse mode and resets the unit to the default conditions.

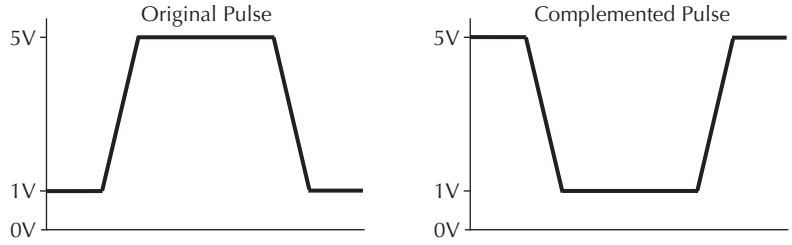
** For details about Segment ARB, refer to User's Manual, Segment ARB waveforms, page 5-6.

Complement mode

Pulse modes: Standard
LPT function: pulse_output_mode

Each channel pulse output can be set for the complement mode. As shown in Figure 11-1, when a pulse is complemented, low pulse goes to the high level, and high pulse goes to the low level.

Figure 11-1
pulse_output_mode



Current limit

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_current_limit

Current limit can be independently set for each pulse card channel. Current limit can be set up to ±200 mA for the 5 V range and up to ±800 mA for 20 V range. Current limit is used to protect the DUT by using the specified DUT load to calculate the voltage required to reach the current limit. A pulse generator channel will not exceed the voltage required to reach the set current limit value at the specified DUT load.

DC Mode

Pulse modes: Standard
LPT function: pulse_dc_output

Each Keithley pulse card channel can be set to output a fixed DC voltage level, rather than pulses. The maximum and minimum output voltage is range dependent. See Pulse high/low for details.

CAUTION **The pulse_vlow and pulse_dc_output commands set the voltage value output by the pulse channel when it is turned on (using pulse_output). If the output is already enabled, these commands will change the voltage level immediately, even before the pulsing is started with a pulse_trig command.**

Full Arb waveform

Pulse modes: Full Arb
LPT function: arb_array

A Full Arb waveform can be defined for each pulse card channel. A Full Arb waveform is made up of user-defined points. A time interval is set to control the time between the waveform points.

The **arb_array** function includes parameters to define the number of points in a waveform (length), the time interval between points (TimePerPt), an array of voltage values (levelArr), and a file name (fname). The array sets the voltage value for each point.

Refer to the User's Manual, Full arb, page 1-29 for additional information and an example of a Full Arb waveform. For more information about the Full Arb waveform example see the User's Manual, Figure 1-24.

Full Arb waveform file

Pulse modes: Full Arb
LPT function: arb_file

A Full Arb waveform that is saved as a `.kaf` file (using the arb_array function) can be loaded for each channel of the pulse card. The **arb_file** function is used to load the file.

Once a Full Arb waveform is loaded, use pulse_output to turn on the appropriate channel, then use pulse_trig to select the trigger mode and start pulse output.

Full Arb waveforms created using KPulse are saved as `.kaf` files, and can be loaded using the arb_file function. Refer to Section 13 for details on using KPulse.

High Endurance Output Relay

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_ssrc

Model 4205-PG2 only – The high endurance output relay (HEOR) is a solid-state relay (SSR) on each channel of the pulse card that can be controlled with the pulse_ssrc LPT function. Note that this setting is independent of the output relay (see pulse_output). A simplified schematic showing the relays is provided in the User's Manual, Figure 1-21.

The pulse_ssrc function is used to control the HEOR as follows:

- **Manual**: Open or close the relay.
- **Auto**: Segment ARB mode controls the relay.
- **Trigger Out Driven**: The relay state will follow the trigger output.

The HEOR is typically controlled via the Segment ARB capability (see seg_arb_define and seg_arb_file).

Pulse count

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_burst_count

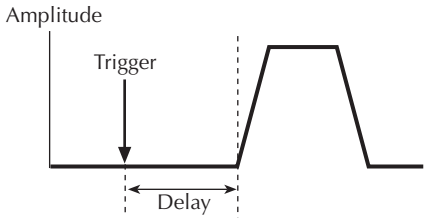
When a burst Trigger mode sequence is triggered, the pulse generator will output the specified number of pulses and then stop. Pulse count can be set from 1 to 2^{32}-1. Burst count can be set independently for each pulse generator channel. The pulse count value takes affect for the selected burst mode when the pulse_trig function is executed.

Pulse delay

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_delay

Pulse delay can be set independently for each pulse generator channel. Pulse delay can be set from 10 ns to (Period - 10 ns). As shown in Figure 11-2, pulse delay is the time from pulse trigger initiation to the rising edge of the pulse.

Figure 11-2
Pulse delay



Pulse halt

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_halt

The pulse output from the pulse card are stopped and both channels are turned off.

Pulse high/low

Pulse modes: Standard
LPT functions: pulse_vhigh and pulse_vlow

Pulse high and pulse low can be set independently for each pulse generator channel. The setting range is dependent on the Source range setting and the Pulse load setting:

CAUTION **The pulse_vlow and pulse_dc_output commands set the voltage value output by the pulse channel when it is turned on (using pulse_output). If the output is already enabled, these commands will change the voltage level immediately, even before the pulsing is started with a pulse_trig command.**

50 Ω load:

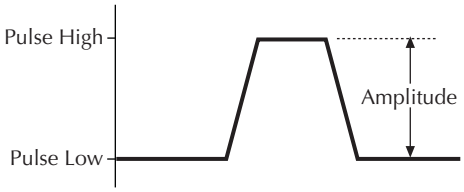
5 V range (high speed): Pulse high and pulse low can be set from -5 V to +5 V.
20 V range (high voltage): Pulse high and pulse low can be set from -20 V to +20 V.

1 M Ω load:

5 V range (high speed): Pulse high and pulse low can be set from -10 V to +10 V.
20 V range (high voltage): Pulse high and pulse low can be set from -40 V to +40 V.

As shown below, pulse low-to-pulse high is the peak-to-peak amplitude of the pulse. The pulse high and pulse low settings take effect immediately during continuous pulse output. Otherwise, the setting take effect when the next pulse trigger is initiated.

Figure 11-3
Pulse low-to-pulse high



Pulse load

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_load

DUT impedance of the load (DUT) can be independently set for each channel. The impedance can be set from 1 Ω to 1 M Ω..

The purpose of setting the DUT load to a value other than 50 Ω is to simplify the calculation of the output levels. An example, if the DUT load is set to 50 Ω, but the actual DUT load is a high impedance of 1 M Ω, setting a voltage level of 2 V will result in a 4 V pulse at the DUT. Setting the DUT load to 1 M Ω will permit the set voltage to match the actual voltage, so setting a 2 V level will result in a 2 V pulse, with the pulse card taking the DUT impedance into account.

Pulse output

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_output

Each channel of the pulse card can be individually controlled (on or off). When the channel is off, the output is in a high-impedance (open) state. After a channel is turned on, pulse output will start when a pulse trigger is initiated. Note that if a Pulse delay has been set, pulse output will start after the delay period expires.

NOTE *It is good practice to routinely turn off the outputs of the pulse card after a test has been completed.*

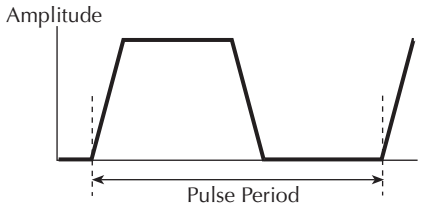
Pulse period

Pulse modes: Standard
LPT function: pulse_period

The setting range for pulse period depends on the selected Source range. For the 5 V range (high speed), pulse period can be set from 20 ns to 1 s. For the 20 V range, pulse period can be set from 500 ns to 1 s. Both channels use the same period, regardless of the source range. This mode is set and takes affect when the pulse trigger is initiated.

As shown below, the pulse period is measured from the start of the rising transition of the pulse to the start of the rising transition of the next pulse.

Figure 11-4
pulse-period



The pulse period setting takes effect immediately during continuous pulse output. Otherwise, the period setting takes effect when the next pulse trigger is initiated.

Pulse trigger output

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_trig_output

By default, output triggers from the pulse card are enabled and available at the Trigger Out connector. The **pulse_trig_output** function is used to control (on or off) output triggers.

With Output Trigger enabled, each output pulse will initiate an output trigger pulse. See Pulse generator card output trigger later in this section for details on output triggers. Figure 11-15 shows the behavior of output triggers (T_O) for the three trigger modes (Continuous, Burst, and Trig Burst).

Pulse trigger source

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_trig_source

An input trigger is used to start pulse output. The **pulse_trig_source** function is used to select the source of the input trigger: software or one of external trigger sources.

With the software trigger source selected, the pulse_trig function is used to select the trigger mode (Continuous, Burst, or Trig Burst) and initiate the start of pulse output. With an external trigger source selected, the pulse_trig function selects the trigger mode and arms pulse output. Pulse output will start when an external trigger pulse is applied to the EXTERNAL IN connector of the pulse card.

There are four External Trigger sources:

- **External, initial trigger only (rising)**: The first rising-edge trigger pulse applied to TRIGGER In will start and control pulse output.
- **External, initial trigger only (falling)**: Same as above, except the initial falling-edge trigger will start and control pulse output.
- **External, trigger per pulse (rising)**: Rising-edge trigger pulses applied to TRIGGER IN will start and control pulse output.
- **External, trigger per pulse (falling)**: Same as above, except falling-edge triggers will start and control pulse output.
- **Internal Trigger Bus**: The internal bus trigger source is used for synchronizing multiple PMU/PGU cards for standard pulse using the legacy pulse functions (pulse_vhigh, pulse_vlow, pulse_width, etc.). This trigger source is used only by the Models 4220-PGU and 4225-PMU.

See External triggering later in this section for details on external trigger sources. Figure 11-17 shows the behavior of the pulse output (P_O) for the various external trigger sources and trigger modes.

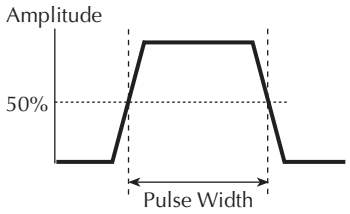
Pulse Width

Pulse modes: Standard
LPT function: pulse_width

Pulse width can be independently set for each pulse generator channel. For the 5 V range (high speed), pulse width can be set from 10 ns to (period - 10 ns). For the 20 V range, pulse width can be set from 100 ns to (Period - 100 ns).

Pulse width is based on the full width, half max method (FWHM). As shown below, the pulse width is measured at the median (50% amplitude) point from the rising edge of the pulse to the falling edge of the pulse.

Figure 11-5
pulse_width



The maximum pulse width that can be set depends on the selected period for the pulse. For example, if the period is set for 500 ns, the maximum pulse width that can be set for the 5 V (high speed) range is 490 ns (500 ns - 10 ns = 490 ns).

The pulse width setting takes effect immediately during continuous pulse output. Otherwise, the width setting takes effect when the next pulse_trig is initiated.

Reset

Pulse modes: Standard, Full Arb, Segment ARB
LPT functions: pulse_init and pg2_init

There are two functions used to reset the pulse generator to the default settings listed in Table 11-1. The **pulse_init** function returns the pulse card to the defaults for the presently selected pulse mode (Standard, Full Arb or Segment ARB). The **pg2_init** function resets the pulse generator to the specified pulse mode and its default conditions.

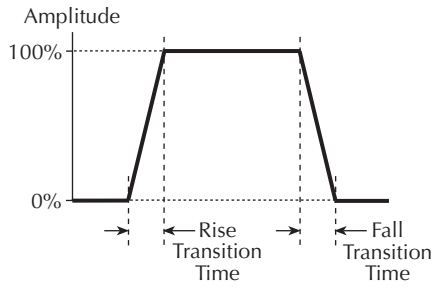
Rise/fall time

Pulse modes: Standard
LPT function: pulse_rise and pulse_fall

Rise time and fall time can be set independently for each pulse generator channel using the **pulse_rise** and **pulse_fall** functions. For the 5 V range (high speed), the minimum rise and fall times can be set at 10 ns. For the 20 V range, the minimum rise and fall times can be set at 100 ns for the Model 4200-PG2 and Model 4205-PG2 cards and 50 ns for the Model 4220-PGU and Model 4225-PMU cards. As shown in Figure 11-6, the pulse rise time and fall time is measured from the 0% and 100% amplitude points on the rising and falling edge of the pulse.

The pulse rise and fall time settings take effect immediately during continuous pulse output. Otherwise, the rise time setting takes effect when the next pulse trigger is initiated.

Figure 11-6
Rise time and Fall time

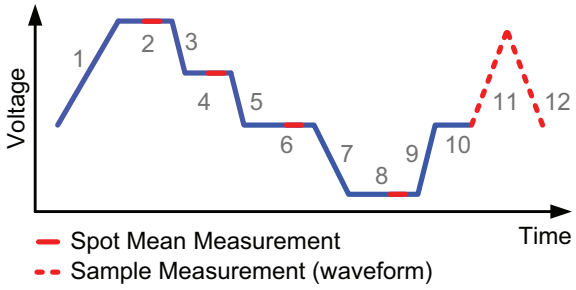


Segment ARB waveform

A Segment ARB® waveform consists of line segments as illustrated in Figure Figure 11-7. The blue line represents the voltage waveform; the red represents the measure windows. Figure 11-7 contains a 12-segment waveform. Two types of measurements are supported, spot mean and sample. Spot mean measurements take a mean of all the samples within the measured window and result in a single reading for that segment. The sample measurement reports all of the samples within the measurement window which can then be plotted versus time to provide a waveform.

There are two types of Segment ARB waveforms, legacy and updated. The legacy Segment ARB mode, defined by seg_arb_define and seg_arb_file, has source only capabilities designed for the Model 4205-PG2 pulse card. Updated Segment ARB capabilities including measurement (for the Model 4225-PMU) and sequences (Model 4225-PMU and Model 4220-PGU).

Figure 11-7
Example of waveform measured using seg_arb_sequence



Pulse Mode: Sequence, Segment ARB

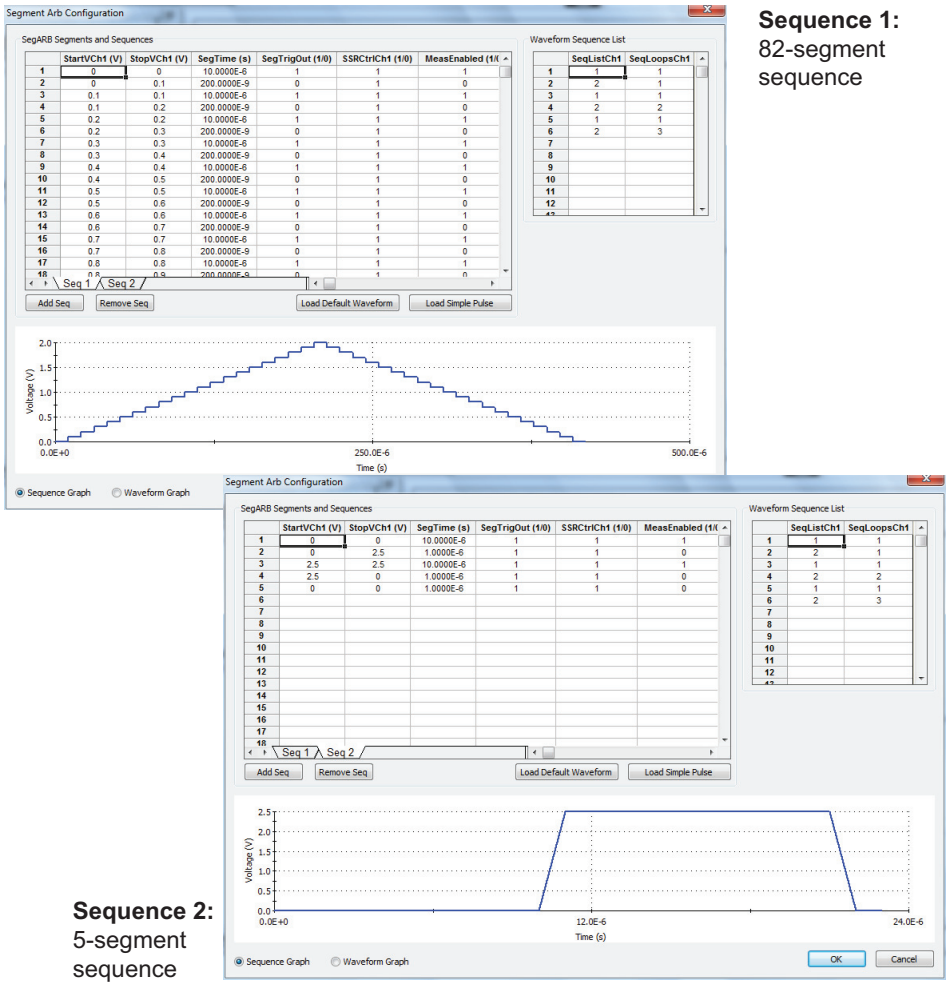
LPT Functions: seg_arb_sequence, seg_arb_waveform

The sequence Segment ARB capability is supported by the Model 4225-PMU and Model 4220-PGU. Only the PMU supports the measurement capability (Figure 11-7). A sequence is a set of three or more segments and a waveform is one or more sequences. Up to 2048 segments can be in a sequence. For a waveform, up to 512 sequences, with a repeat, or a loop count of 1 to 10^12. The sequences are useful for stress/measure reliability sequences. For example, one sequence is the stress portion (DC or AC type) and another sequence is the measurement. The measurement could be a single spot mean, a sweep, a waveform capture, or a combination. The looping capability allows for easy addition of time to subsequent stress intervals.

In the following figure, sequence 1 is an 82 segment sequence, additional rows are not shown in the figure (see Figure 11-8). Sequence 2 is a 5 segment sequence. There are additional columns to the right for per-segment measurement configuration.

Definition showing two sequences with looping is illustrated in Figure 11-9. The graph of the waveform measurement of a two-sequence Segment ARB with looping, from UTM PMU_SegArb_Complete in the PMU-DUT-Examples project (in the _Pulse project folder), is shown in Figure 11-10.

Figure 11-8
Two sequence definitions



Sequence 1: 82-segment sequence

Sequence 2: 5-segment sequence

Figure 11-9
Definition showing two sequences with looping

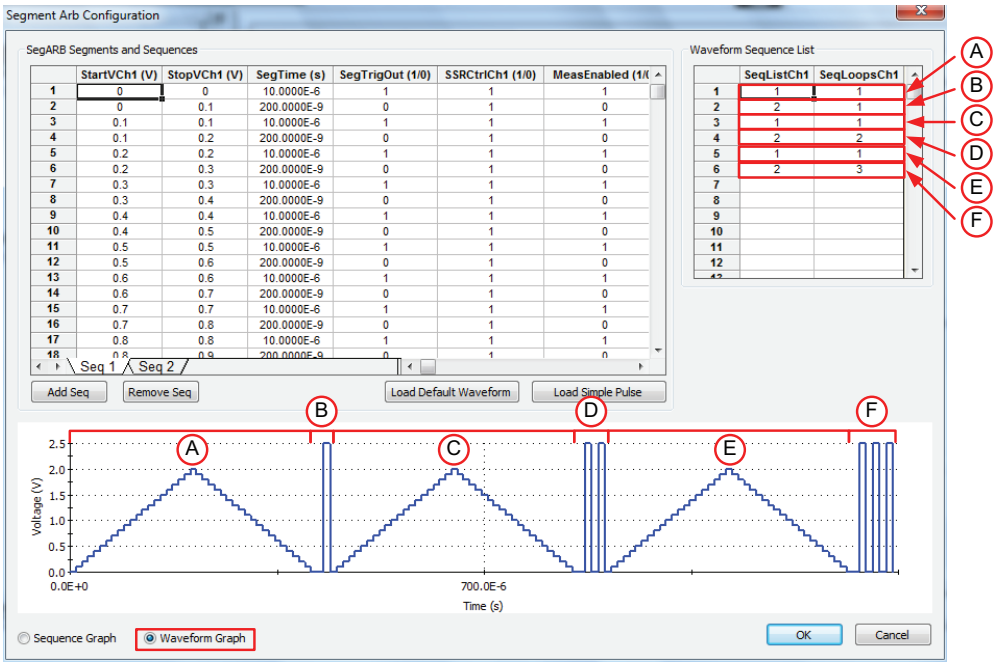
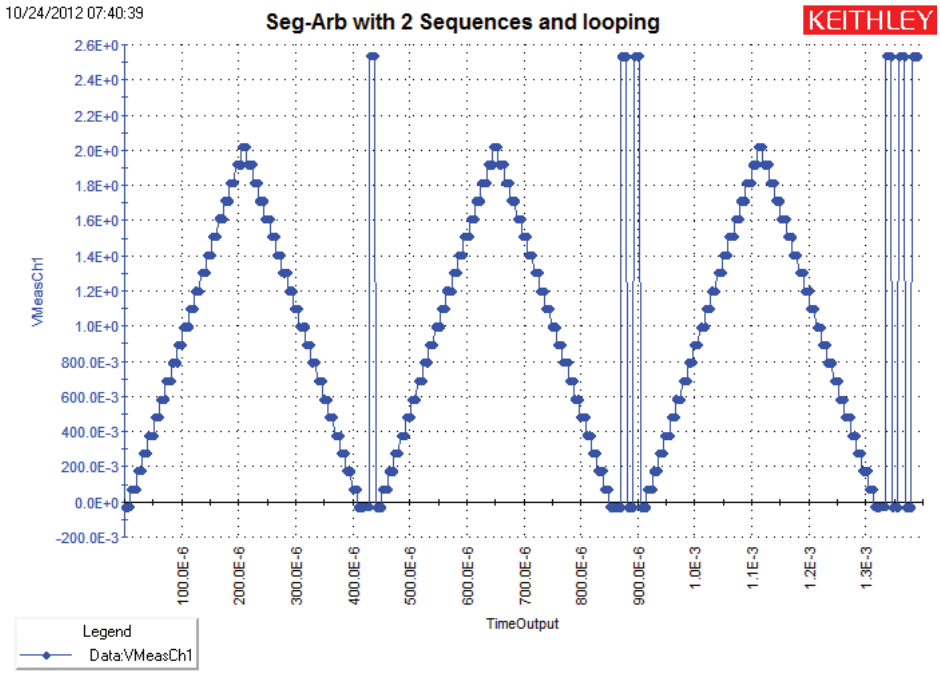


Figure 11-10
Graph of the waveform (two-sequence with looping) measurement



Pulse modes: Source, Segment ARB
LPT function: seg_arb_define

Each channel of the Model 4502-PG2 pulse card can be configured to output its own unique Segment ARB® waveform. A Segment ARB waveform is made up of user-defined segments (up to

1024). Each segment can have a unique time interval, start value, stop value, output trigger level (TTL high or low), and output relay state (open or closed).

The **seg_arb_define** function includes parameters to set the number of segments (*nSegments*) that make up the waveform, and arrays for start values (*startvals*), stop values (*stopvals*), time values (*timevals*), trigger values (*triggervals*), and output relay states (*outputRelayVals*).

Refer to Segment ARB earlier in this section for additional information and an example of a source only Segment ARB® waveform (see the User's Manual, Figure 1-23).

Segment ARB waveform file

Pulse modes: Source only, Segment ARB
LPT function: seg_arb_file

A Segment ARB waveform that is saved as a .ksf file can be loaded for each channel of the pulse card. The **seg_arb_file** function is used to load the file.

Once a Segment ARB waveform is loaded, use pulse_output to turn on the appropriate channel, and then use pulse_trig to select the trigger mode and start pulse output.

Segment ARB waveforms created using KPulse are saved as .ksf files, and can be loaded using the seg_arb_file function. Refer to Section 13 for details on using KPulse.

Source range

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_range

Source range can be set independently for each pulse generator channel. There are two ranges for the output level: 5 V and 20 V (into a 50 Ω DUT load). Selecting the 5 V range also selects high speed pulse output. For the 5 V high speed range, the Pulse period can be as short as 20 ns and Pulse Width can be set as short as 10 ns. This setting takes effect when the next pulse trigger is initiated.

Trigger mode

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_trig

With the software trigger source selected, the **pulse_trig** function is used to select the trigger mode (Continuous, Burst or Trig Burst) and initiate the start of pulse output. With an external trigger source selected, the **pulse_trig** function is used select the trigger mode and arm pulse output. Pulse output will start when an external trigger is applied to the EXTERNAL IN connector. The pulse_trig_source function is used to select the trigger source. Refer to Triggering later in this section for details.

In the continuous trigger mode, the pulse card will output pulses continuously until the pulse_halt or pulse_output (turned off) functions are called. In a burst mode, it will output the programmed number of pulses and then stop. This setting affects both output channels.

If Pulse delay is set to zero, pulse output will start immediately when triggered. If pulse delay is >0, pulse output will start after the delay period expires.

Trigger polarity

Pulse modes: Standard, Full Arb, Segment ARB
LPT function: pulse_trig_polarity

The pulse_trig_polarity function is used to set the polarity (positive or negative) for Model pulse card trigger output pulses of the pulse card. Trigger output provides a TTL level output that is at the same frequency (period) as the pulse card output channels, but has a 50% duty cycle.

Trigger output is used to synchronize pulse output with the operations of a scope card or other external instruments (for example, other pulse cards). When connected to the scope card, each trigger pulse from the pulse card will trigger a scope waveform measurement. Ensure that triggering for the pulse card and scope card is set for the same polarity. Refer to Triggering later in this section for details.

DUT resistance determines pulse voltage across DUT

Each output channel of the pulse card is effectively a voltage source with a series 50 Ω resistance. Since the pulse card, like many other pulse sources, does not have any measurement capability, the actual voltage across the device under test (DUT) is directly related to the DUT resistance.

There are many output effects that are part of using a pulse generator. This section covers the most common issue, which is the voltage across the DUT not meeting initial expectations.

There is more than one way to explain the effect of impedance on the DUT voltage. The approach used below relies on DC concepts and explains the settled portion of the pulse and does not require knowledge or use of RF concepts. RF concepts are necessary to explain time-based effects, such as rise time, overshoot, ringing and reflection.

Figure 11-11
Schematic of pulse channel and connected DUT.

As shown in Figure 11-11, the voltage across the DUT, V_{DUT}, is directly related to the resistance R_{DUT}. The discussion here is simplified, including resistive impedances only.

Looking at Figure 11-11, the 50 Ω output resistance of the pulse channel and the DUT resistance effectively make a voltage divider. Ohm's law is the only concept required to determine I_{DUT} and V_{DUT} in this simple, non-dynamic, approach. First calculate the current, and then use the current to calculate the voltage across the DUT.

Example 1: Ideal situation (50 Ω DUT), shown in Figure 11-12.

R_{DUT}	50 Ω
Pulse Vhigh	5 V
Pulse Vlow	0 V
Pulse Load	50 Ω

Figure 11-12
5 V pulse into a 50 Ω DUT load

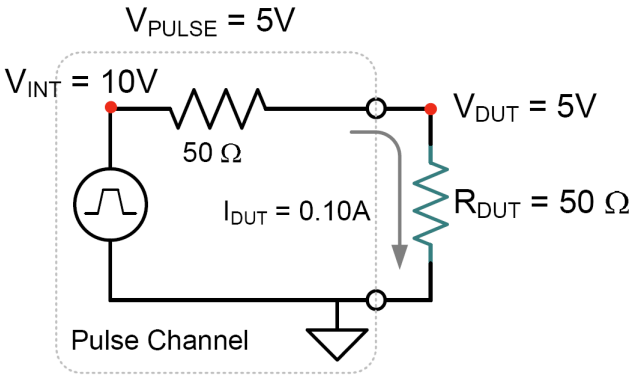


Figure 11-12: Example 1, showing a 5 V pulse into a 50 Ω DUT load.

V = I * R Ohm's Law

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50 + 50 Ω) = 10 V / 100 Ω = 0.1 A$

Note that the internal voltage, V_{INT}, is 2x the requested 5 V so that V_{DUT} = 5 V. The 2x multiplier is the default case, where R_{DUT} = Pulse Load = 50 Ω. Pulse load is a channel-based parameter that allows the pulse generator to calculate the value of the multiplier to source the proper voltage to provide the desired pulse level for the supplied value of the pulse load. Whenever the value for pulse load does not equal the actual DUT resistance, the voltage across the DUT will not match the programmed voltage level.

Calculate V_{DUT}:

$V_{DUT} = I_{DUT} * R_{DUT} = 0.1 A * 50 Ω = 5 V.$

Example 2: High Resistance DUT, shown in Figure 11-13

R_{DUT}	1 MΩ
Pulse Vhigh	5 V
Pulse Vlow	0 V
Pulse Load	50 Ω

Figure 11-13
5 V pulse into a 1 MΩ DUT load

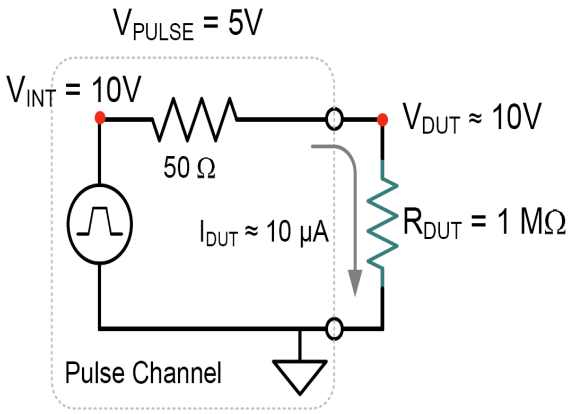


Figure 11-13: Example 2, showing a 5 V pulse into a 1 MΩ DUT load.

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50 + 1\ M\Omega) = 10\ V / 1.00005E+6\ \Omega = 9.9995E-6\ A \approx 10\ \mu A$.

Calculate V_{DUT}:

$V_{DUT} = I_{DUT} * R_{DUT} = 10\ \mu A * 1E6\ \Omega \approx 10\ V$.

The reason V_{DUT} ~ 10 V is the pulse load = 50 Ω, but R_{DUT} = 1 MΩ. The pulse generator calculates the output voltage based on the pulse load, and uses a 2x multiplier to determine V_{INT} = 10 V. Since the pulse generator cannot measure the output voltage or DUT load, this example results with a V_{DUT} that 2x the programmed level.

Example 3: Low Resistance DUT

R_{DUT}	5 Ω
Pulse Vhigh	5 V
Pulse Vlow	0 V
Pulse Load	50 Ω

Figure 11-14
5 V pulse into a 5 Ω DUT load

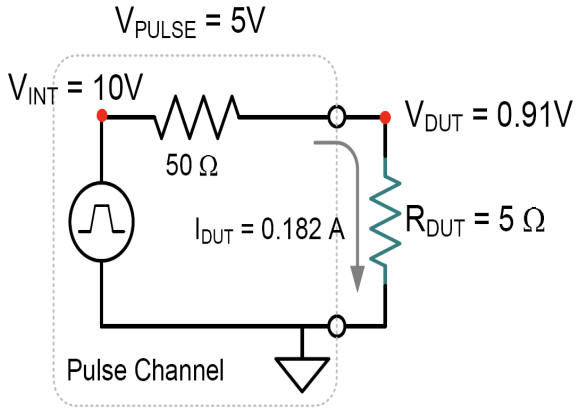


Figure 11-14: Example 3, showing a 5 V pulse into a 5 Ω DUT load.

Calculate the current, I_{DUT}:

$I_{DUT} = V_{TOTAL}/R_{TOTAL} = V_{INT}/ (50 + 5\ \Omega) = 10\ V / 55\ \Omega = 0.182\ A$

Calculate V_{DUT}:

$V_{DUT} = I_{DUT} * R_{DUT} = 0.182\ A * 1E6\ \Omega = 0.91\ V.$

All pulse parameters were constant across all three examples. Only the DUT load (R_{DUT}) was changed, resulting in a range of V_{DUT} from less than 1V to nearly 10 V, with corresponding changes in I_{DUT}. This variation is related to the constant pulse load = 50 Ω. Reprogramming the pulse load value to match R_{DUT} would prevent this behavior, but is not practical in many situations.

This effect is inherent to the pulse generator and DUT system. The general approach for calculating the current and voltage is given in the examples above. Calculating the maximum voltage and current available for any given DUT resistance is similar to the above examples. The only difference is that the maximum voltage available is used for the range. For the High Speed Range, V_{MAX} = 10 V, for the High Voltage range, V_{MAX} = 40 V.

Example 4: Maximum Voltage and Current, High Speed Range

$V = I * R$

$V_{MAX} = 10\ V$ for High Speed Range

$R_{DUT} = 5\ \Omega$

$I_{MAX} = V_{MAX} /\ /\ R_{TOTAL} = V_{MAX} / (R_{pulse\ generator} + R_{DUT})$

$I_{MAXDUT} = 10\ V / (50\ \Omega + 5\ \Omega) = 10 / 55 = 0.182\ A$

$V_{MAXDUT} = I_{MAX} * R_{DUT} = 0.182\ A * 5\ \Omega = 0.91V$

These I_{MAX} and V_{MAX} values match the values in Table 1 for the 5 Ω row. Note that these calculations do not incorporate any cabling or interconnect losses, that may range from 1 Ω up to 10-20 Ω, depending on the interconnect method and DUT type (packaged or on-wafer).

Example 5: Maximum Voltage and Current, High Voltage Range

$V = I * R$

$V_{MAX} = 40\ V$ for High Voltage Range

$R_{DUT} = 250\ \Omega$

$I_{MAX} = V_{MAX} /\ /\ R_{TOTAL} = V_{MAX} / (R_{pulse\ generator} + R_{DUT})$

$I_{MAXDUT} = 40\ V / (50\ \Omega + 250\ \Omega) = 40 / 300 = 0.133\ A$

$V_{MAXDUT} = I_{MAX} * R_{DUT} = 0.133\ A * 250\ \Omega = 33.3V$

These I_{MAX} and V_{MAX} values match the values in Table 2 for the 250 Ω row. These calculations do not incorporate any cabling or interconnect losses, that may range from 1 Ω up to 10-20 Ω, depending on the interconnect method and DUT type (packaged or on-wafer).

Table 11-2 and Table 11-3 below show the maximum current and voltage for various R_{DUT} values for the two output ranges available on each pulse card channel. Note the calculated current and voltage values do not include any cabling or interconnect losses, that reduce both the maximum available V_{DUT} and I_{DUT}.

Table 11-2
Available I & V for the High Speed (5 V) Range of the Keithley pulse card

Pulse Card High Speed (5 V) Range – Maximum I and V vs Resistance		
Resistance (Ω)	Voltage (V)*	Current (A)*
1	0.196	0.196
5	0.909	0.182
10	1.667	0.167
25	3.333	0.133
50	5	0.100
100	6.667	0.067
250	8.333	0.033
1 k	9.524	0.0095
10 k	9.950	0.000995

NOTE ** Approximate value, does not account for interconnect losses.*

Table 11-3
Available I & V for the High Voltage (20 V) Range of the Keithley pulse card

Pulse Card High Voltage (20 V) Range – Maximum I and V vs Resistance		
Resistance (Ω)	Voltage (V)*	Current (A)*
1	0.784	0.784
5	3.636	0.727
10	6.667	0.667
25	13.333	0.533
50	20	0.400
100	26.667	0.267
250	33.333	0.133
1 k	38.095	0.038
10 k	39.801	0.00398

NOTE **Approximate value, does not account for interconnect losses.*

Because V_{DUT} is a function of the load applied to the pulse card channel, this effect is sometimes called the Load Line Effect.

Load Line Effect Compensation: Coping with the Load Line Effect

There are 3 common ways of working with this effect. The simplest one is to program the DUT load into the pulse card channel using the pulse_load command (see Section 8 for the pulse card commands), or setting the Pulse Load value in the Kpulse virtual front panel (see Section 5 of the

User's Manual). The pulse card will calculate the appropriate V_{INT} to output so that the V_{DUT} pulse waveform, specified by pulse_vlow and pulse_vhigh, has the desired levels. This works well for high impedance devices or device terminals (R_{DUT} = 1 k Ω), such as the gate terminal on a CMOS field effect transistor (FET). Unfortunately, many times R_{DUT} is not known or varies. A key example of a varying R_{DUT} is the drain-source resistance during a V_D-I_D sweep, where R_{DS} is changing from point-to-point and sweep-to-sweep.

There is basically only one way to handle this situation, with two different levels of implementation. In general, assume the DUT is a FET. If the test consists of a single, or limited number of gate and drain, test points, the necessary voltages can be determined by pre-characterizing each unique set of test conditions.

This pre-characterization requires some way to measure the pulse heights, which is typically done using an oscilloscope and an iterative trial and error approach. Each desired test voltage needs to be measured, with the pulse levels adjusted until the desired voltage is reached. Record each pulse level required to reach the desired V_{DUT} levels.

An alternate method requires some type of software algorithm, in addition to the oscilloscope. The software, such as a 4200 user test module (UTM), would take V_{DUT} readings then determine the appropriate adjustments to reach the desired V_{DUT}. This is also an iterative approach, but the software algorithm would perform the iterations and handle all the measurements and calculations. This type of software algorithm approach is used in the optional pulse IV packages available for the 4200-SCS, such as the 4200-PIV-A and 4200-PIV-Q. Note that the algorithms in these packages are not available for general purpose use and have been customized for the source/measure approach and system interconnect unique to each package.

NOTE *Basic Oscilloscope hardware and concepts are covered in the User's Manual, SCP2 (Oscilloscope), page 1-32.*

NOTE *The Model 4225-PMU has built-in load line effect compensation. For details, see Load line effect compensation (LLEC) for the PMU*

kiscopeulib UTM descriptions

The kiscopeulib UTMs control either the Model 4200-SCP2HR or 4200-SCP2 scope cards. The modules contained in the KIScope user library are listed in Table 11-4 with detailed information following the table.

NOTE *The scope control is by a usrlib, not within LPTLib. LPTLib commands such as devclr and devint do not apply to the scope card. To initialize the scope card, use init_kiscope.*

Table 11-4
kiscopeulib UTMs

User Module	Description
autocal_kiscope	Performs an autocal on the scope card
close_kiscope	Closes the VISA session connection to the scope card
downrange_kiscope	Sets one channel of the scope vertical voltage range to the next lower value (unless the range is already at the minimum value)
gethandle_kiscope	Obtains a VISA session connection handle to the scope card
getrange_kiscope	Returns the proper Y scale (voltage range) on the scope card
getreading_kiscope	Retrieves a reading from one channel of the scope card
init_kiscope	Sets up the scope card. This module should be run before any of the other routines
meas_kiscope	Gets an averaged gated single reading from one channel of the scope card
readwaveform_kiscope	Captures and retrieves a waveform from one channel of the scope card
set_kiscope	Sets a variety of scope card parameters
uprange_kiscope	Sets one channel of the scope vertical voltage range to the next higher value (unless the range is already at the maximum value)

autocal_kiscope

Description Although this UTM is not associated with a particular application, this routine should be run before any pulse IV calibration or compensation routine. The autocal_kiscope performs an autocal on the scope card. Refer to Tables 11-5 and 11-6 for inputs and outputs, respectively.

Procedure Follow the prompt and disconnect all cables from the scope card.

Table 11-5
Inputs for autocal_kiscope

Input	Type	Description
none	NA	NA

Table 11-6
Outputs for autocal_kiscope

Output	Type	Description
Temperature	double *	The internal scope card temperature in °C.
AutoCalStatus	long *	The autocal status. 0 = OK, 1 = Failed, 2 = Corrupt calibration

close_kiscope

Description The close_kiscope routine closes the VISA session connection to the scope card. Refer to Tables 11-7 and 11-8 for inputs and outputs, respectively.

The companion command is gethandle_kiscope.

Procedure This routine should be placed after the last required scope command in a UTM. Note that if a session is not closed, subsequent UTMs using the scope card will not be able to execute.

Table 11-7
Inputs for close_kiscope

Input	Type	Description
none	NA	NA

Table 11-8
Outputs for close_kiscope

Output	Type	Description
none	NA	NA

downrange_kiscope

Description The downrange_kiscope routine sets one channel of the scope vertical voltage range to the next lower value. This routine is used, along with uprange_kiscope, to provide a software-based auto-range capability. If the scope channel is already at the lowest range, then no change will be made. Note that the available ranges are dependent on the channel impedance settings. Refer to Tables 11-9, 11-10, and 11-11 for inputs, outputs, and return values, respectively.

Companion routines: uprange_kiscope

NOTE *Note that the decision to change ranges is not made in this routine.*

Procedure Call this routine to change the scope range down one range.

Table 11-9
Inputs for downrange_kiscope

Input	Type	Description
channel	int	The scope card channel to downrange (1 or 2)

Table 11-10
Outputs for downrange_kiscope

Output	Type	Description
none	NA	NA

Table 11-11
Return values for downrange_kiscope

Value	Type	Description
present range	Double	New range value, after downrange

gethandle_kiscope

Description The gethandle_kiscope routine obtains a VISA session connection handle to the scope card. This routine is used to establish a connection and also retrieve the handle of an existing connection. Refer to Tables 11-12, 11-13, and 11-14 for inputs, outputs, and return values, respectively.

The companion command is close_kiscope.

Procedure This routine should only be called once an init_kiscope has been run. Note that gethandle_kiscope (listed in this section) is not needed in order to use the user module functions (the gethandle_kiscope routine is already incorporated). However, if the user wants to communicate with the scope card directly, a session handle will be needed.

Table 11-12
Inputs for gethandle_kiscope

Input	Type	Description
none	NA	NA

Table 11-13
Outputs for gethandle_kiscope

Output	Type	Description
none	NA	NA

Table 11-14
Return values for gethandle_kiscope

Value	Description
session	(int) session handle. A null session implies that a scope card was not found

getrange_kiscope

Description The getrange_kiscope routine returns the proper Y scale (voltage range) on the scope card to use for a given voltage and impedance. Refer to Tables 11-15, 11-16, and 11-17 for inputs, outputs, and return values, respectively.

NOTE *The scope range is the entire voltage span centered around 0. For example, the 5 V range is -2.5 to +2.5 volts.*

Procedure Call this routine to determine the proper range to use:

- **Model 4200-SCP2HR**: At 50 Ω, the maximum voltage range is 10 Vpp (±5 V), and at 1 M Ω, the maximum voltage range is 50 Vpp (±25 V).
- **Model 4200-SCP2**: At 50 Ω, the maximum voltage range is 10 Vpp (±5 V), and at 1 M Ω, the maximum voltage range is 300 Vpp (±150 V).

Note that if the voltage is too high, the maximum voltage range is returned.

Table 11-15
Inputs for getrange_kiscope

Input	Type	Description
value	double	Voltage level
load	double	Input impedance of the scope channel, either 50 Ω (50) or 1 M Ω (1 E6)

Table 11-16
Outputs for getrange_kiscope

Output	Type	Description
none	NA	NA

Table 11-17
Return values for getrange_kiscope

Value	Description
Scope range	Range required to measure the queried voltage and given input impedance.

getreading_kiscope

Description The getreading_kiscope routine retrieves a reading from one channel of the scope card. Before using this command, the scope must be configured for a measurement type using the set_kiscope routine (refer to the set_kiscope routine description for more information). Refer to Tables 11-18, 11-19, and 11-20 for inputs, outputs, and return values, respectively.

Procedure Set the desired measurement type (see above) before calling this command for the desired channel. This command returns a single value; to return the entire waveform, use readwaveform_kiscope.

Table 11-18
Inputs for getreading_kiscope

Input	Type	Description
channel	int	Scope card channel (1 or 2)

Table 11-19
Outputs for getreading_kiscope

Output	Type	Description
reading	double *	The scope reading.

Table 11-20
Return values for getreading_kiscope

Value	Description
0	No errors
-12001	Error – Scope not initialized
-12002	Error – Invalid scope channel
-12003	Error – Scope measurement type setup invalid
-12004	Error – Scope reading failed
-12005	Error – Scope not found in the system
-12006	Error – Scope invalid average number
-12007	Error – Unable to malloc memory space
-12008	Error – Scope read waveform failed
-12009	Error – Invalid trigger mode
-12010	Error – Invalid scope impedance
-12011	Error – Invalid sub function
-12013	Error – Invalid rate
-12014	Error – Invalid number of points
-12015	Error – Invalid value for autoscale Y

init_kiscope

Description The init_kiscope routine sets up the scope card as follows:

1. Initializes the scope structure

2. Configures the card for the following default settings:
   - Enable both channels
   - Impedance is set to 1 M Ω
   - Sampling is set to 1GHz with number of points set to 200
   - Y scale/range is set to 10 V (-5 V to +5 V)
   - Trigger is set to external

Refer to Tables 11-21, 11-22, and 11-23 for inputs, outputs, and return values, respectively. This routine should be used whenever `devint` is called (`devint` is a LPTLib function call).

Procedure This routine should be the first one called in order to use the scope.

Table 11-21
Inputs for init_kiscope

Input	Type	Description
none	NA	NA

Table 11-22
Outputs for init_kiscope

Output	Type	Description
none	NA	NA

Table 11-23
Return values for init_kiscope

Value	Description
0	No errors
-12005	Error – Scope not found in the system

meas_kiscope

Description The meas_kiscope routine gets a reading from one channel of the scope card. This command uses the getreading_kiscope command, and adds averaging of multiple readings and software-based auto-ranging of the scope Y scale. Before making a measurement with this command, the scope must be appropriately configured using set_scope (refer to set_kiscope description for more details).

Refer to Tables 11-24, 11-25, and 11-26 for inputs, outputs, and return values, respectively.

Procedure Specify desired channel for reading along with the desired averaging.

Table 11-24
Inputs for meas_kiscope

Input	Type	Description
channel	int	Scope card channel (1 or 2)
avgnum	int	Number of readings to average (1 to 1000)

Table 11-25
Outputs for meas_kiscope

Input	Type	Description
data	double *	The scope reading

Table 11-25
Outputs for meas_kiscope

Input	Type	Description
usedrange	double *	The range (Y Scale) used for the reading

Table 11-26
Return values for meas_kiscope

Value	Description
0	No errors
-12001	Error – Scope not initialized
-12002	Error – Invalid scope channel
-12003	Error – Scope measurement type setup invalid
-12004	Error – Scope reading failed
-12005	Error – Scope not found in the system
-12006	Error – Scope invalid average number
-12007	Error – Unable to malloc memory space
-12008	Error – Scope read waveform failed
-12009	Error – Invalid trigger mode
-12010	Error – Invalid scope impedance
-12011	Error – Invalid sub function
-12013	Error – Invalid rate
-12014	Error – Invalid number of points
-12015	Error – Invalid value for autoscale Y

readwaveform_kiscope

Description The readwaveform_kiscope routine captures and retrieves a waveform from one channel of the scope card. Before using this command, the scope must be configured using set_scope (refer to set_kiscope description for more details). Refer to Tables 11-27, 11-28, and 11-29 for inputs, outputs and return values, respectively.

Procedure Set up the scope using set_kiscope and call the routine to retrieve the waveform. This command returns the entire waveform; to return a single value, use getreading_kiscope.

Table 11-27
Inputs for readwaveform_kiscope

Input	Type	Description
channel	int	Scope card channel (1 or 2)
Time_size	int	Size of the time array. Should match Waveform_size
Waveform_size	int	Size of the waveform array; should match Time_size

Table 11-28
Outputs for readwaveform_kiscope

Output	Type	Description
Time	double *	Array of time values (s)
Waveform	double *	Array of readings (V)

Table 11-29
Return values for readwaveform_kiscope

Value	Description
0	No errors
-12001	Error – Scope not initialized
-12002	Error – Invalid scope channel
-12003	Error – Scope measurement type setup invalid
-12004	Error – Scope reading failed
-12005	Error – Scope not found in the system
-12006	Error – Scope invalid average number
-12007	Error – Unable to malloc memory space
-12008	Error – Scope read waveform failed
-12009	Error – Invalid trigger mode
-12010	Error – Invalid scope impedance
-12011	Error – Invalid sub function
-12013	Error – Invalid rate
-12014	Error – Invalid number of points
-12015	Error – Invalid value for autoscale Y

set_kiscope

Description The set_kiscope routine sets a variety of scope parameters. Refer to Tables 11-30, 11-31, 11-32, and 11-33 for inputs, subfunctions, outputs, and return values, respectively.

Procedure Choose the appropriate subfunction and specify the parameters.

NOTE *Note that not all subfunctions require 2 parameters. Therefore, pad all single-parameter functions with a 0 for the second parameter.*

Table 11-30
Inputs for set_kiscope

Input	Type	Description
subfunction	int	Subfunctions are listed in Table 11-31.
param1	double	First parameter
param2	double	Second parameter

Table 11-31
Subfunctions for set_kiscope

Subfunction	Parameters
AUTOSET_SCP	The scope will pick appropriate parameters based on the input signals.
ACQMODE_SCP	Selects continuous or single sequence: param1: Sequence – 0 (RUNSTOP_SCP) or 1 (SINGLESEQ_SCP) param2: Not used – pass in 0
MEASTYPE_SCP	Measurement type: param1: Type – AMPL_SCP, HIGH_SCP, LOW_SCP, RMS_SCP, MEAN_SCP param2: Channel – 1 or 2
XSCALE_SCP	Horizontal scale: param1: Rate – 1.25E9, 1E9, 500E6, 200E6, 100E6, 50E6, 20E6, 10E6, 1E6, 500E3, 200E3, 100E3, 50E3, 20E3, 10E3 param2: Points – 1 to 1048576
YSCALE_SCP	Vertical scope range (the closest appropriate range based on which scope load is chosen): param1: Range param2: Channel – 1 or 2
AUTOSCALEY_SCP	Turns autoscaling on or off: param1: 0 (off) or 1 (on) param2: Channel – 1 or 2
XOFFSET_SCP	x scale offset: param1: Offset param2: Not used – pass in 0
YOFFSET_SCP	y scale offset: param1: Offset – Maximum is (0.5 * YSCALE) param2: Not used – pass in 0
AVGNUM_SCP	Number of waveforms to average: param1: Average number param2: Not used - pass in 0
TRIGMODE_SCP	Trigger method: param1: Source – EXT_SCP, CH1_SCP, CH2_SCP param2: Level
TRIGLEV_SCP	Sets trigger level (V): param1: Level param2: Channel – 1 or 2
TRIGPOSITION_SCP	Sets horizontal (time) position of the trigger (as a percentage of acquisition time): param1: 0 to 1 (see Note) param2: Not used – pass in 0
DELAY_SCP	Delay after trigger (s): param1: Delay param2: Not used – pass in 0
IMPED_SCP	Scope input impedance (Ω): param1: Impedance (in ohms) – 50 or 1 E6 param2: Channel – 1 or 2
GATE_SCP	Gated or cursor measurement (AVGNUM is not used for gated measurements): param1: 0 (not gated) or 1 (gated) param2: Not used - pass in 0
CURPOSX_SCP	Set cursor positions for gated measurements: param1: Cursor 1 param2: Cursor 2

Note The percentage is based on the sample rate and number of points. For example, if the sample rate is 200MS/s and number of points is 200, it will take 1µs to acquire the 200 readings. To set the trigger position to 40% (400 ns), you would pass 0.4 in the function.

Table 11-32
Outputs for set_kiscope

Output	Type	Description
none	NA	NA

Table 11-33
Return values for set_kiscope

Value	Description
0	No errors
-12001	Error – Scope not initialized
-12002	Error – Invalid scope channel
-12003	Error – Scope measurement type setup invalid
-12004	Error – Scope reading failed
-12005	Error – Scope not found in the system
-12006	Error – Scope invalid average number
-12007	Error – Unable to malloc memory space
-12008	Error – Scope read waveform failed
-12009	Error – Invalid trigger mode
-12010	Error – Invalid scope impedance
-12011	Error – Invalid sub function
-12013	Error – Invalid rate
-12014	Error – Invalid number of points
-12015	Error – Invalid value for autoscale Y

uprange_kiscope

Description The uprange_kiscope routine sets one channel of the scope vertical voltage range to the next higher value. This routine is used, along with downrange_kiscope, to provide a software-based auto-range capability. If the scope channel is already at the highest range, then no change will be made. This routine is dependent on the channel impedance. Refer to Tables 11-34, 11-35, and 11-36 for inputs, outputs, and return values, respectively.

Companion routines: downrange_kiscope

NOTE *Note that the decision to change ranges is not made in this routine.*

Procedure Call this routine to change the scope range up one range.

Table 11-34
Inputs for uprange_kiscope

Input	Type	Description
channel	int	The scope card channel to uprange (1 or 2)

Table 11-35
Outputs for uprange_kiscope

Output	Type	Description
none	NA	NA

Table 11-36
Return values for uprange_kiscope

Value	Type	Description
present range	double	New range value, after uprange

Triggering

Triggering is used for the following operations:

- Basic triggering: Configure the Keithley pulse card for the desired trigger mode (Continuous, Burst or Trig Burst), and use a software trigger to start pulse output.
- Pulse-measure synchronization: Synchronize the pulse-measure operation of a pulse generator card and the scope card. When pulse output starts, the scope is triggered by the pulse generator to capture one or more waveforms.
- Pulse output synchronization: Use external triggering to synchronize the pulse output of multiple pulse generator cards.

NOTE *Details on the trigger functions discussed in the following paragraphs are provided in the LPT Library Function Reference in Section 8.*

Basic triggering

Pulse output of a pulse card can be started by a software trigger. The LPT function for the software trigger also sets the trigger mode. Enabled pulse generator channels will then output a continuous string of pulses (continuous trigger mode), or a burst of pulses (burst or trig burst trigger mode).

Software trigger source

There are two types of trigger sources: Software and External. In order to use a software trigger to start pulse output, the software trigger source must first be selected. The pulse_trig_source function is used to select the software trigger source. Refer to External triggering for details on the external trigger sources.

Trigger mode

With the software trigger source selected, using the pulse_trig function selects one of the following trigger modes and initiates (triggers) the start of pulse output:

- **Continuous**: This trigger mode continuously outputs pulses when pulse output is started.
- **Burst** or **Trig Burst**: Either of these trigger modes output a burst of pulses when pulse output is started. A burst is a finite number of pulses (1 to 2^{32}-1). The only difference between burst and trig burst is the behavior of trigger output (refer to Pulse generator card output trigger).

NOTE *When using the burst or trig burst trigger mode, make sure to first set the pulse count before starting pulse output. The pulse_burst_count function is used to set the burst count.*

Example LPT function sequence: Basic triggering

The following LPT function sequence uses the software trigger to initiate a 3-pulse burst for both channels, where both the pulse and trigger output three pulses:

```
// Stop pulse generator output:
pulse_halt(VPU1);

// Select software trigger source:
pulse_trig_source(VPU1, 0);

// Set channel 1 for a burst count of 3:
pulse_burst_count(VPU1, 1, 3);

// Turn channel 1 on:
pulse_output(VPU1, 1, 1);

// Select the trigger burst mode and trigger start of burst output:
pulse_trig(VPU1, 2);
```

Pulse-measure synchronization

The scope card can be synchronized to the pulse output of a Model 4205-PG2 pulse generator. When pulse output for the Model 4205-PG2 starts, it will trigger the scope to capture one or more waveforms or readings.

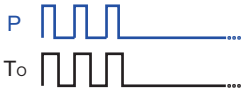
Pulse generator card output trigger

With output trigger enabled, an output pulse will initiate a TTL level, 50% duty cycle output trigger pulse. The trigger pulses are available at the TRIGGER OUT connector of the pulse generator card.

The pulse_trig_output function is used to enable or disable output trigger. Output trigger can be set for positive (rising edge) or negative (falling edge) polarity. Use the pulse_trig_polarity function to set the polarity of output trigger.

Figure 11-15 shows the behavior of output triggers (T_O) for the three trigger modes. Notice that for the Burst mode, output triggers continue even though pulse output has stopped. For the trig burst mode, output triggers will stop when pulse output stops.

Figure 11-15
Pulse generator card output trigger

Trigger Mode	Standard Pulse	Full Arb Pulse	Segment Arb Pulse*
Continuous			
Burst			
Trig Burst			

P = Pulse Output
T_O = Trigger Output
*Segment Arb has user defined trigger output (0 or 1) for each segment.

Scope card ext trig

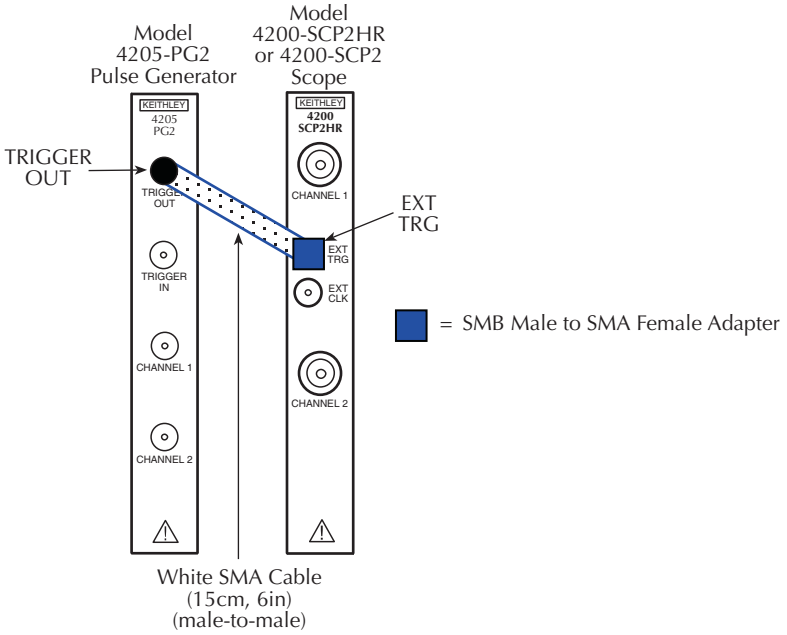
For pulse-measure synchronization, the scope must be set for the normal sweep mode, and set to be triggered by the leading-edge or falling edge trigger from the Model 4205-PG2 pulse generator.

Refer to the ZTEC scope User's Manual for details on triggering. Triggers from the pulse generator are to be applied to the EXT TRIG connector of the scope card.

Trigger connections: pulse-measure

As shown Figure 11-16, TRIGGER OUT of the pulse generator card is connected to Trg Ext of the scope card. The SMA cable and adapter for the scope card are supplied items.

Figure 11-16
Trigger connections for pulse-measure synchronization



Example LPT function sequence: pulse-measure synchronization

The following LPT function sequence uses the software trigger to initiate a 3-pulse burst for CHANNEL 1. The three output pulses will trigger the scope to perform three measurements. It assumes the scope is configured to trigger on leading edge triggers from the pulse generator:

```
// Stop pulse generator output:
pulse_halt(VPU1);

// Turn off trigger output:
pulse_trig_output(VPU1, 0);

// Set the output trigger polarity to positive:
pulse_trig_polarity(VPU1, 1);

// Select Software trigger source:
pulse_trig_source(VPU1, 0);

// Set channel 1 for a burst count of 3:
pulse_burst_count(VPU1, 1, 3);

// Turn channel 1 on:
pulse_output(VPU1, 1, 1);

// Select the Trig Burst trigger mode and trigger start of burst output. Each pulse will trigger the
// scope to perform a measurement.
pulse_trig(VPU1, 2);
```

Pulse output synchronization

External triggering can be used to synchronize the pulse outputs of multiple pulse generator cards. The master pulse card can trigger itself and the other pulse cards to simultaneously start their pulse outputs.

NOTE *The Model 4200-PG2 does not support input triggering. Therefore, it cannot be used for pulse output synchronization.*

External triggering

With a pulse card configured for external triggering, an external trigger source applied to the TRIGGER IN connector will control pulse output. The pulse_trig_source function is used to select one of the following four external trigger sources (refer to Figure 11-17):

- **External, initial trigger only (rising)**: The first rising-edge trigger pulse applied to Trigger In will start pulse output. For the Continuous trigger mode, the initial trigger pulse will start continuous pulse output. For the Burst and Trig Burst modes, the initial trigger pulse will start the pulse burst.

  In this mode, the pulse generator card will only look for the initial trigger. After this initial trigger, all subsequent pulses output are gated to the internal pulse generator clock. For continuous output, or high count burst output, the synchronization across pulse generator cards will be gradually worsen. For deterministic pulse-to-pulse timing across cards, use the Trigger per Pulse mode.
- **External, initial trigger only (falling):** Same as above, except a falling-edge trigger will start pulse output.
- **External, trigger per pulse (rising)**: Rising-edge trigger pulses applied to Trigger In will start and control pulse output. For the Continuous trigger mode, each trigger pulse will initiate one output pulse. If input triggers stop, pulse output will stop. Operation for Burst and Trig Burst is similar, except pulse output will stop after the last pulse of the burst is triggered (additional input triggers are ignored).
- **External, trigger per pulse (falling):** Same as above, except falling-edge triggers will start pulse output.

NOTE *Models 4220-PGU and 4225-PMU: Do not use the two external **falling** trigger sources with the positive trigger output polarity (see pulse_trig_polarity function) on the master card that triggers itself and other slave cards. These two falling trigger sources should only be used when an external piece of equipment is being used to supply the trigger pulses to the Models 4220-PGU and 4225-PMU. This applies to all three pulse modes (Standard Pulse, Segment ARB and Full Arb).*

NOTE *For the master pulse generator card, the polarity of the pulse trigger source and pulse trigger polarity (pulse_trig_polarity function) must be the same. If using a rising-edge trigger source, the pulse trigger polarity must be positive. If using a falling-edge trigger source, the pulse trigger polarity must be negative.This requirement applies to all three pulse modes (Standard Pulse, Segment ARB and Full Arb).*

NOTE *When triggering multiple Model 4205-PG2 cards in a master slave configuration, changing the master trigger output polarity (pulse_trig_polarity function) will result in a*

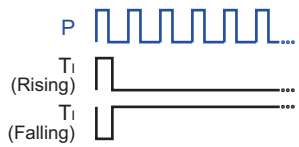
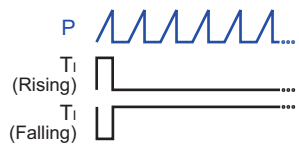
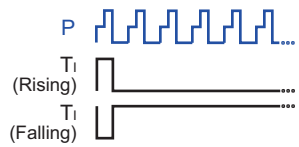
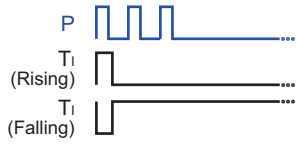
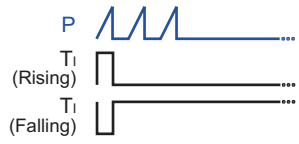
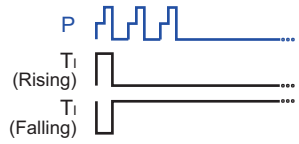
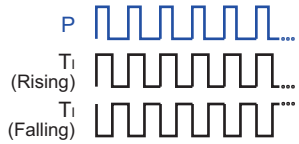
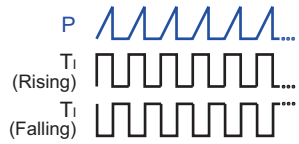
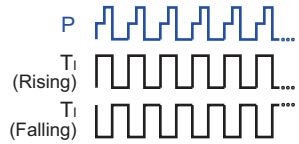
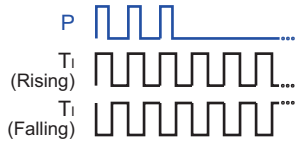
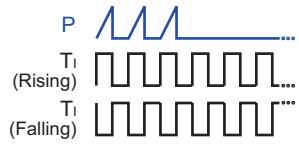
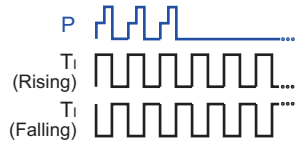
---

transition in the trigger output levels that may be interpreted as a trigger pulse by the other cards.

NOTE *Triggering transition time needs to be <100 ns. Trigger output Level and Trigger In Level need to be TTL.*

After selecting an External trigger source, use the pulse_trig function to select the trigger mode (Continuous, Burst, or Trig Burst) and arm pulse output of the pulse generator card.

Figure 11-17
Pulse generator card trigger input (external triggering)

Trigger Source	Trigger Mode	Standard Pulse	Full Arb Pulse	Segment Arb Pulse*
External, Initial Trigger Only	Continuous			
	Burst & Trig Burst			
External, Trigger per Pulse	Continuous			
	Burst & Trig Burst			

P = Pulse Output
T_I = External trigger Input (rising or falling)

* Segment ARB trigger output is user defined and is customized to match the application. See Multi-channel synchronization with the Segment ARB Mode for information related to synchronization of multiple pulse Segment ARB® waveforms

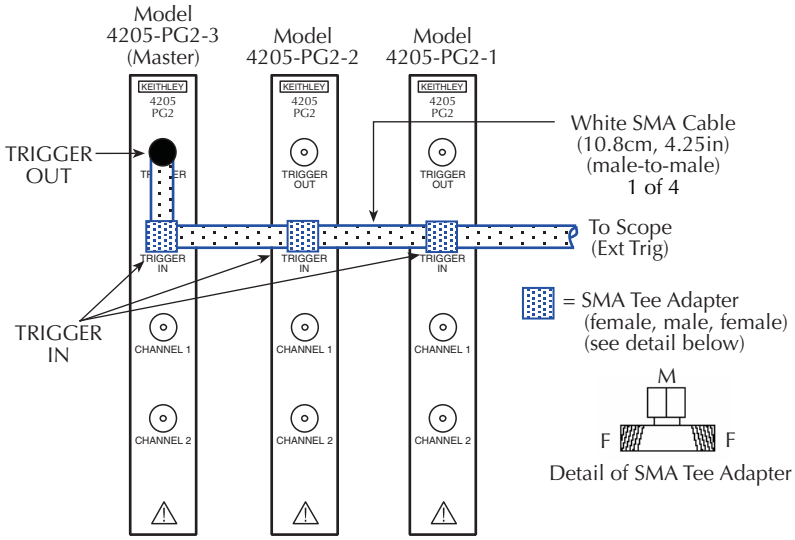
Trigger connections: Output synchronization

To synchronize the start of pulse output for multiple pulse generator cards, TRIGGER OUT of one card can be used to trigger itself and the other cards. As shown in Figure 11-18, TRIGGER OUT of one pulse generator card is connected to Trigger In of all the pulse generator cards. Figure 11-18 also shows that TRIGGER OUT can also be used to trigger another instrument, such as the scope card.

In Figure 11-18, the trigger cable configuration shows that Model 4205-PG2-3 is the master pulse generator. These connections provide optimum timing for triggering. When trigger output is enabled for Model 4205-PG2-3, it will trigger itself and the other pulse generators to start pulse output. It will also trigger the scope.

NOTE *Figure 11-25 shows a connection drawing for output synchronization that includes the scope card connection.*

Figure 11-18
Trigger connections for pulse output synchronization of multiple Model 4205-PG2s



Example LPT function sequence: pulse output synchronization

The following LPT function sequence will simultaneously start continuous pulse output for the three Model 4205-PG2s shown in Figure 11-18. It assumes that Model 4205-PG2-1 is already set for positive polarity output triggers.

// Stop pulse generators' output:
```
pulse_halt(VPU1);
pulse_halt(VPU2);
pulse_halt(VPU3);
```

// Turn off trigger outputs:
```
pulse_trig_output(VPU1, 0);
pulse_trig_output(VPU2, 0);
pulse_trig_output(VPU3, 0);
```

// Set the output trigger polarity to positive:
```
pulse_trig_polarity(VPU1, 1);
```

// Select External – Initial Trigger Only (Rising) trigger sources:
```
pulse_trig_source(VPU1, 1);
pulse_trig_source(VPU2, 1);
pulse_trig_source(VPU3, 1);
```

// Enable the channel 1 outputs:
```
pulse_output(VPU1, 1, 1);
pulse_output(VPU2, 1, 1);
pulse_output(VPU3, 1, 1);
```

// Arm (waiting for external trigger) pulse generators and select the Continuous trigger mode:
```
pulse_trig(VPU1, 1);
pulse_trig(VPU2, 1);
pulse_trig(VPU3, 1);
```

// Start Trigger Output for Model 4205-PG2-3. This will start the synchronized pulse output:
```
pulse_trig_output(VPU3, 1);
```

Multi-channel synchronization with the Segment ARB Mode

There are two methods for utilizing the Segment ARB capability, depending on the number of channels that are synchronized. Each method uses a different trigger-in source mode, specified by the pulse_trig_source function.

When using Segment ARB for 1 or 2 channels on a single card, no trigger cabling is required and the trigger source is set to Software. See Figure 11-19 (8-segment waveform) and the corresponding definition in Table 11-37. Note that the trigger is controllable on a segment-by-segment basis and will be output via the Trigger Out connector.

Figure 11-19
An example of a Segment ARB waveform and its associated trigger output waveform

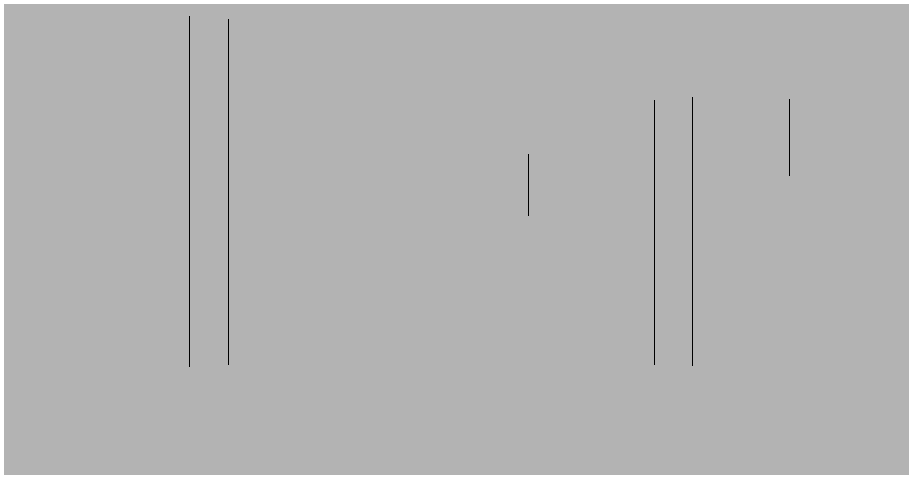


Table 11-37
Segment ARB definition with trigger definition for intra-waveform synchronization, as shown in Figure 11-19

Segment	Start V	Stop V	Time	Trig/Sync	HEOR
1	0	0	250µs	0	1
2	0	8	1µs	1	1
3	8	8	5µs	0	1
4	8	0	50µs	0	1
5	0	0	300µs	0	1
6	0	-10	1µs	1	1
7	-10	-10	5µs	0	1
8	-10	0	1µs	0	1

To synchronize multiple channels across the pulse cards, trigger cabling, trigger-in mode and trigger definition are changed from the single card method. For trigger cabling, one card is used as the master, with its trigger output connected to its own trigger input, as well as all other trigger inputs on the other cards (see Figure 11-18).

The trigger in source mode is set to Trigger per Pulse. The timing across cards may drift up to 0.02%. This drift means that a set of synchronized waveforms, each with multiple pulses, will eventually drift with respect to each other. This is only an issue if the pulse shapes or relative timing is negatively affected by the small amount of drift. For example, 0.02% over a 100µs

segment means a drift of 20 ns. This drift may be an issue if a set of waveforms consists of two pulses, a long pulse followed by a relatively short pulse, as shown in Figure 11-19. If all channels have a similar 2-pulse waveform, the concern may be to synchronize the transition at the start of the second pulse. The first pulse has a width of 5 ms, so the potential drift in the time, across multiple waveforms, could be a maximum of 5 ms x 0.02% = 1µs. This 1µs drift would cause the second pulse of one waveform to be shifted, either earlier or later, relative to another waveform.

For certain applications this drift may be unacceptable, which is the use case for the Trigger per Pulse Trigger-In setting. This setting is also available in the Standard Pulse mode, but is slightly different in that the trigger output is already defined and occurs only once per period or waveform. This Trigger per Pulse mode is sufficient for Standard Pulse mode, where there is only one pulse per waveform, or period. However, the Segment ARB mode permits multiple pulses or shapes within a single waveform, and the waveform period could be quite long and require that the waveforms are synchronous across multiple pulse cards.

To keep the drift of longer, multi-pulse waveforms to an acceptable level, the Trigger per Pulse mode has a unique feature, which utilizes the segment trigger definition to create cross-channel synchronization events. There are two requirements for using this intra-waveform synchronization capability. The first is that the trigger defined in the Segment ARB® waveform must be the same across all channels that are synchronized. Second, there must be at least one or more segments where there is a transition from 0 to 1. For example, if the first segment has a trigger state of 1, then the last segment of the waveform must have a trigger state of 0. Figure 11-19 shows two triggers, which will be used as synchronization events at each time the waveform is output. Note that both trigger input and trigger output must be set to the same polarity, either both positive (rising), or both negative (falling).

The waveform in Figure 11-20 shows the effect of synchronizing across the pulse card. The waveform is still defined as given in Figure 11-19 and Table 11-37. The Trigger-In source is Trigger per Pulse.

Figure 11-20
Waveform effect of synchronizing across PG2 card

NOTE *Figure 11-20 does not depict the 560 ns delay from the trigger-in to pulse output, which occurs when synchronizing multiple pulse cards with trigger cabling and the Trigger per Pulse mode for trigger-in source. For multi-card synchronization, the*

delay is experienced by all channels, and does not cause synchronization problems across PG2 channels (not shown here to simplify the diagram).

Figure 11-20 shows that the trigger definition is now used to define which segments must be synchronized. In this case, all channels will wait during segments 1 and 5, so that all pulse card begin segments 2 and 6 synchronously. The delay caused by the supplied cabling and PG2 circuitry provides 120 ns typical delay, which is added to segments 1 and 5. All channels wait, outputting the last voltage level specified for the segment, until all are ready to start the next segment. For longer waveforms, or waveforms with more time-critical segments, use additional trigger/synchronization events to keep all channel outputs synchronous. The 120 ns typical delay occurs for each synchronization event, which extends the period for each waveform by120 ns times the number of synchronization events.

Pulse source-measure connections

Figure 11-21 provides connector identification for the pulse generator and scope cards.

NOTE *For 4200-PIV-A test connections, see Section 3 4200-PIV-A test connections of the User's Manual. For 4200-PIV-Q test connections, see the Interconnect Assembly Procedure in the 4200-PIV-Q App Note.*

NOTE *TRIGGER OUT of a pulse generator card can be connected to TRIGGER IN of the scope to provide synchronization between pulse output and scope measurements. TRIGGER OUT of a pulse generator card can also be connected to TRIGGER IN of other pulse generator cards to synchronize the start of pulse outputs. For details on using the trigger connectors, refer to Triggering earlier in this section.*

All connection hardware for the pulse generator card and scope card is supplied. The cables used for SMU connections are supplied with the ordered SMUs and preamps.

The following paragraphs cover some fundamental connection schemes for various test configurations using pulse generator cards (Model 4205-PG2) and a digital storage oscilloscope (Model 4200-SCP2HR or 4200-SCP2).

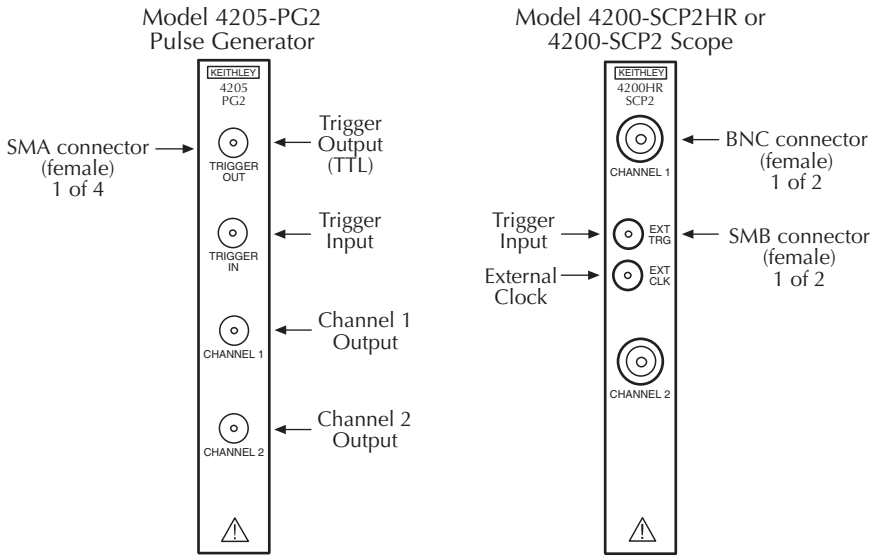
- Pulse generator connections (Figure 11-22)
- Scope connections (Figure 11-23)
- Pulse generator and scope connections (Figure 11-24)
- Multiple pulse generator and scope connections (Figure 11-25)

When using specific pulse or pulse IV applications, refer to Section 4 of the Model 4200-SCS Applications Manual for application-specific connections.

NOTE *To achieve optimum performance, only use the cables, connectors, and adapters that are included with Keithley Instruments pulse source or measure kits.*

WARNING ***For the pulse source-measure configurations, ensure the Model 4200-SCS high voltage is disabled. This will prevent a safety hazard that could result in possible injury or death because of SMU voltages greater than 42V being applied to the device under test or fixture. To disable Model 4200-SCS high voltage, refer to Safety interlock connections in Section 4.***

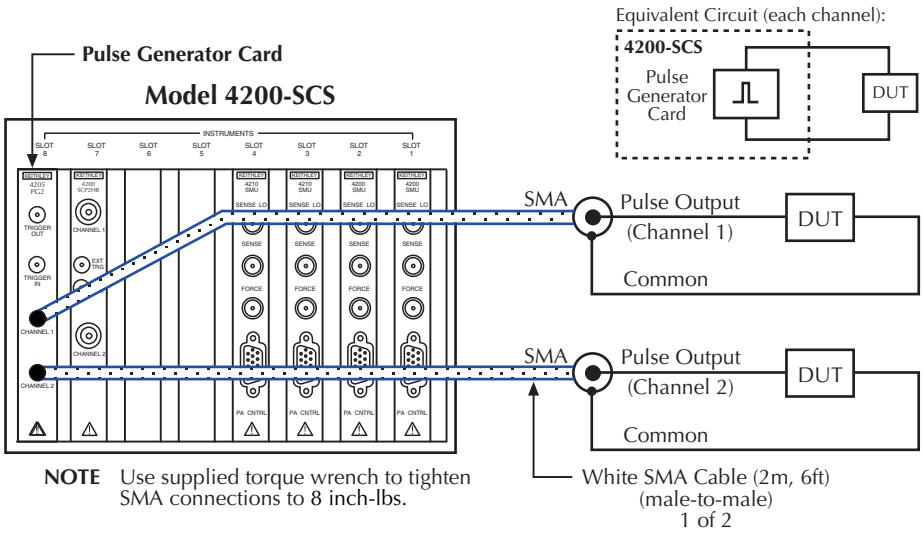
Figure 11-21
Pulse generator and scope card connectors



Pulse generator connections

Figure 11-22 shows a system that uses basic 2-channel pulse generator connections to DUTs.

Figure 11-22
Basic pulse generator connections

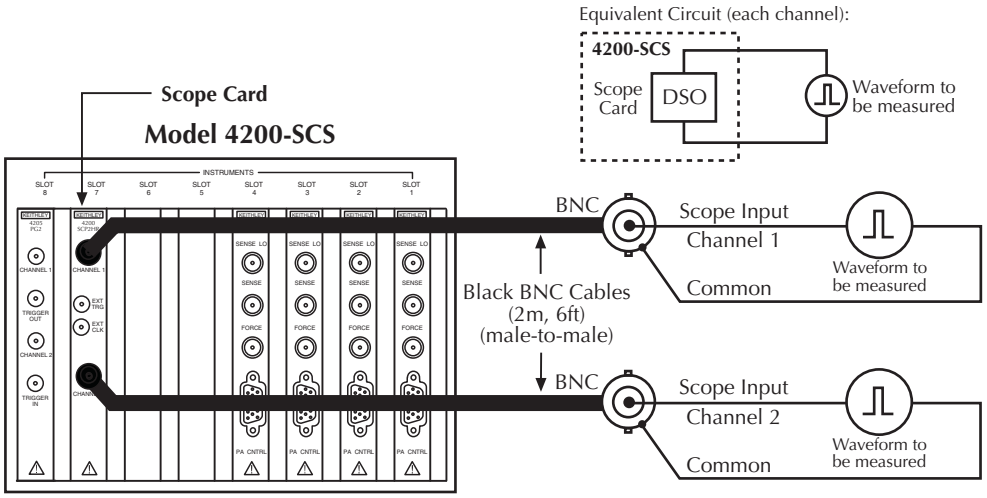


Scope connections

Figure 11-23 shows basic 2-channel scope card connections to the signals to be measured.

NOTE *The BNC cable supplied with the scope card is a 50 Ω cable. When using this cable with the scope be sure that the scope, input is set to the 50 Ω mode. Failure to do so could cause incorrect measurement.*

Figure 11-23
Basic scope connections



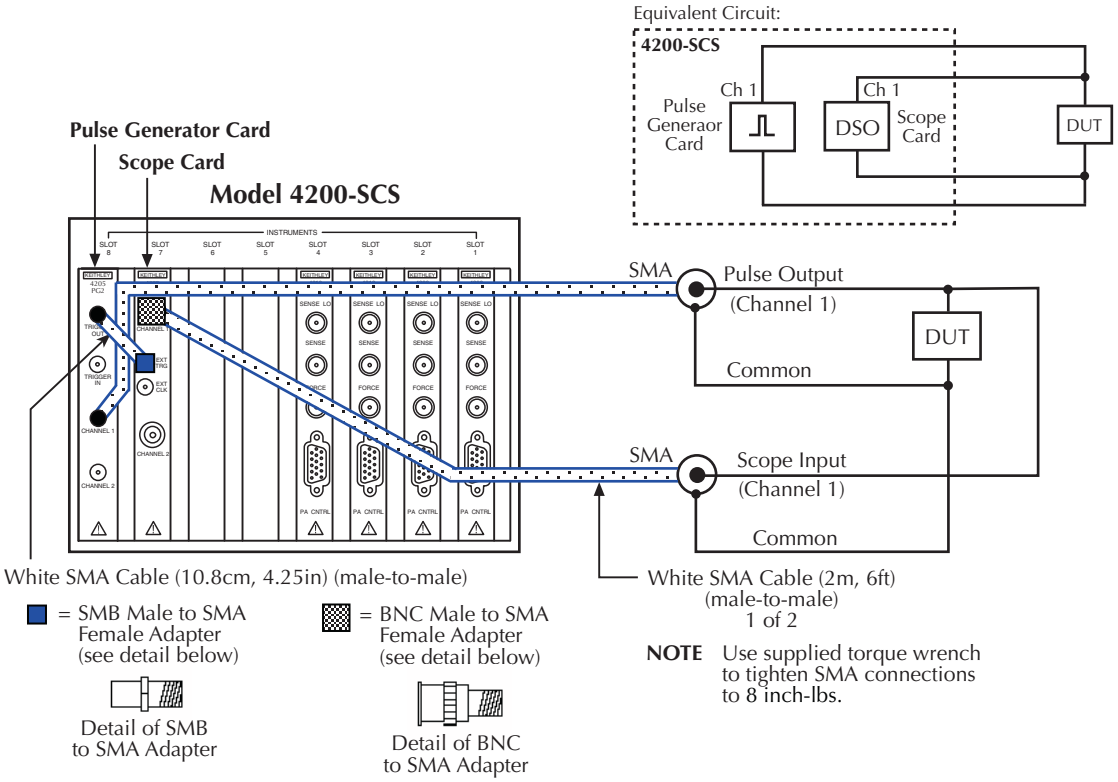
Pulse generator and scope connections

Figure 11-24 shows a connection scheme using both the pulse generator card and the scope card. This configuration uses one channel of the pulse generator card and one channel of the scope card. The second channel of the pulse generator card and scope card can also be used in a similar manner. In this connection scheme, the scope channel input impedance should be set to 1 M Ω, to allow measurement of the voltage across the DUT.

NOTE *The connection scheme in Figure 11-24 uses an SMA cable for the scope input channel. A BNC cable (supplied with the scope) can be used instead. A BNC cable (male-to-male) mates directly to a scope input channel.*

Pulse-measure synchronization: Waveform measurements by the scope card can be synchronized to the pulse outputs of the pulse generator cards. This trigger synchronization is achieved by connecting TRIGGER OUT of a pulse generator card to EXT TRIG of the scope card. In Figure 11-24, the TRIGGER OUT of the pulse generator card is connected to EXT TRIG of the scope card. See Pulse-measure synchronization earlier in this section for details on pulse-measure synchronization.

Figure 11-24
Basic pulse generator and scope connections



Multiple pulse generator and scope connections

Up to three pulse generator cards can be used in a test system. Figure 11-25 shows a connection scheme using three pulse generator cards and a scope card. This example provides pulse output to the gate, drain and source terminals of a FET. Using multiple pulse generator cards allow different pulse periods to be used for their respective pulse outputs. The two channels of the scope are used to capture the waveforms of the voltage across the gate and drain of the FET (as opposed to measuring the current through each DUT pin).

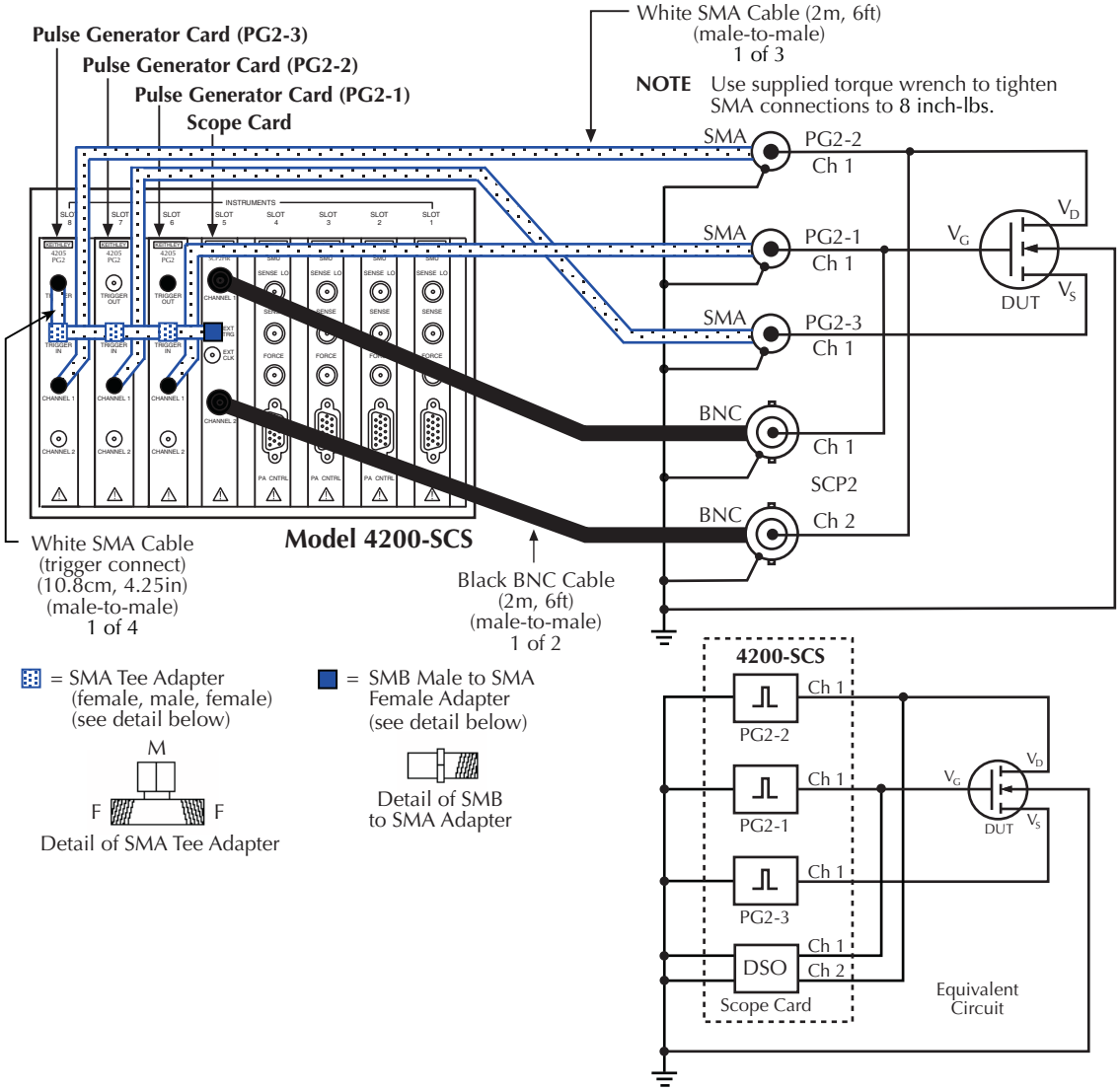
NOTE *The connection scheme in Figure 11-25 uses BNC cables for the scope input channels. SMA cables can be used instead. Figure 11-24 shows how to connect an SMA cable to a scope input.*

Pulse output synchronization: The outputs of the three pulse generator cards can be synchronized to start at the same time. In Figure 11-25, short SMA cables and SMA Tee adapters are used to connect TRIGGER OUT of Model 4205-PG2-3 (making it the master PG2) to TRIGGER IN of the three Model 4205-PG2s. When the trigger output of Model 4205-PG2-3 is enabled (on), it will trigger itself and the other two Model 4205-PG2s to simultaneously start their outputs. See Pulse output synchronization earlier in this section for details on pulse output synchronization.

Pulse-measure synchronization: Waveform measurements by the scope card can be synchronized to the pulse outputs of the pulse generator cards. This trigger synchronization is achieved by connecting TRIGGER OUT of a pulse generator card to EXT TRIG of the scope card. In Figure 11-25, the TRIGGER OUT of Model 4205-PG2-3 is routed to EXT TRIG of the scope

card. See Pulse-measure synchronization earlier in this section for details on pulse-measure synchronization.

Figure 11-25
Multiple pulse generators and scope connection (example configuration)



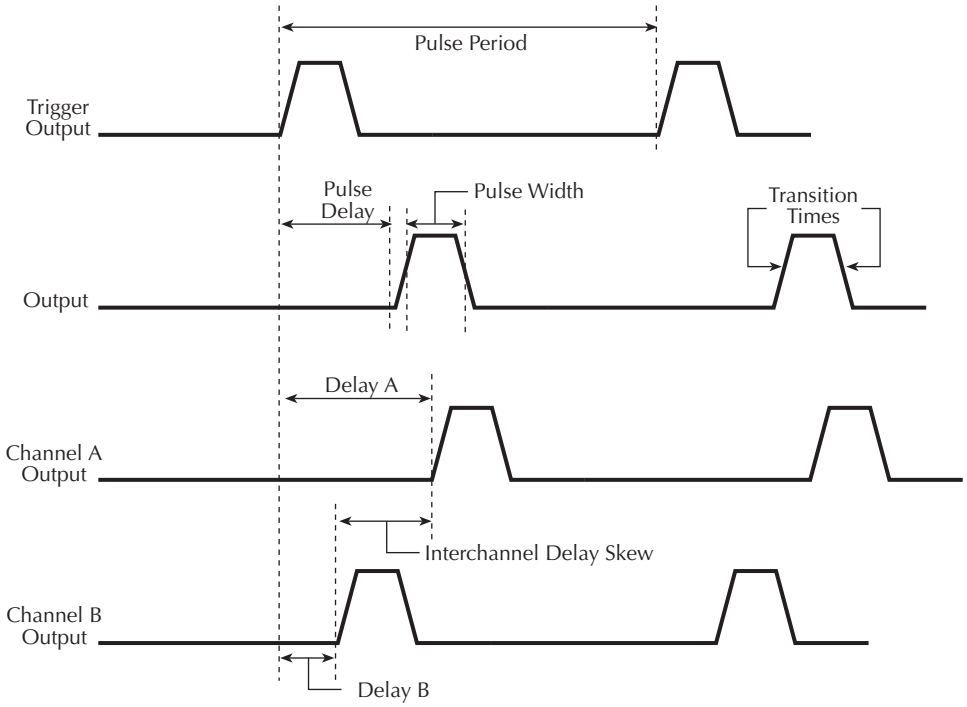
Pulse parameter definitions

The following paragraphs define pulse parameters, which are listed in alphabetical order as follows:

Distortion (preshoot, overshoot, and ringing)
Duty cycle
Interchannel delay (skew)
Jitter
Linearity (deviation)
Pulse delay
Pulse levels
Pulse period
Pulse width
Repeatability
Settling time (rise and fall time)
Transition time

Figure 11-26 provides an overview of the pulse parameters.

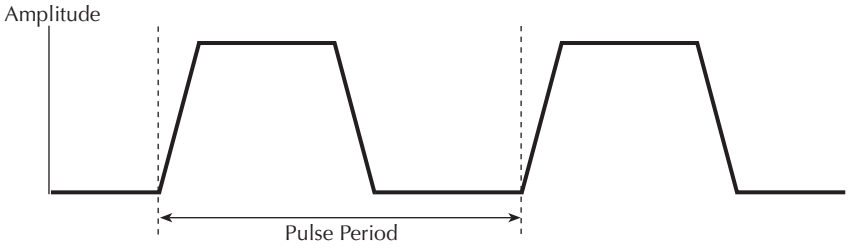
Figure 11-26
Overview of pulse parameters



Pulse period

As shown in Figure 11-27, the pulse period is the time interval between the start of the rising transition edge of consecutive output pulses.

Figure 11-27
Pulse period



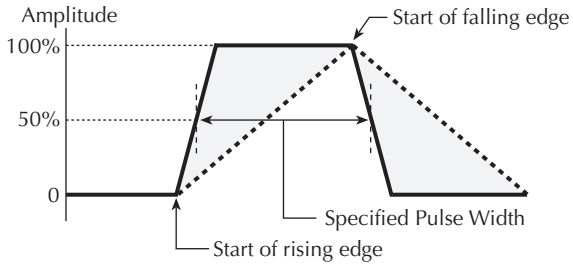
Pulse width

As shown in Figure 11-28, pulse width is the interval between rising-edge and falling-edge medians. For a pulse with the fastest edges (short Transition time) the pulse width is essentially equal to the interval from the start of the rising edge to the start of the falling edge.

By design, the rise and fall times are independently set. Changing the transition time will not change the pulse width. In practice, start points may shift with changes in transition time.

The dash-line pulse in Figure 11-28 shows how increased transition time can affect the rising edge and falling edge of the pulse. The shaded areas of the pulse show the changes in the slopes. Notice that the pulse width interval is not affected. If the transition time is increased any further, the pulse would not reach its programmed amplitude (100%).

Figure 11-28
Pulse width



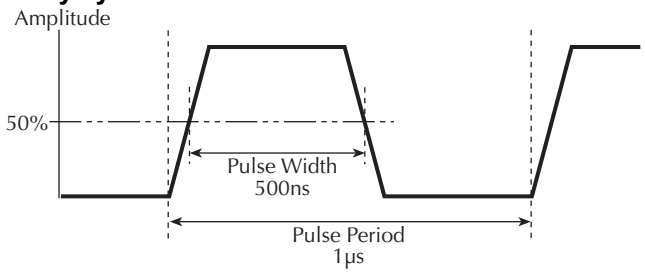
Duty cycle

The duty cycle is the amount of time, as a percentage of the pulse period, that the pulse is on (pulse width). Duty cycle (as a percentage) is calculated as follows:

Duty cycle = (pulse width / pulse period) x 100

Figure 11-29 shows an example for duty cycle.

Figure 11-29
Example of 50% duty cycle

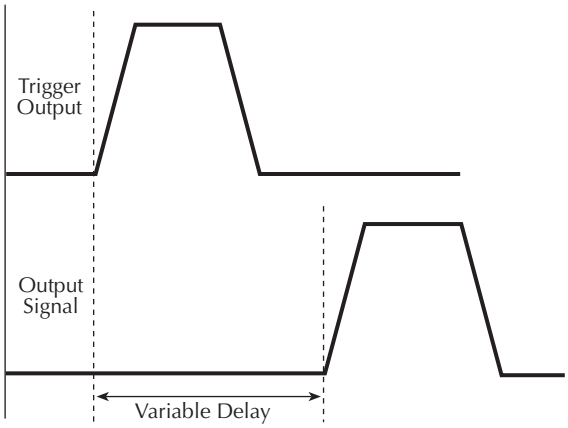


Duty Cycle = (Pulse Width / Pulse Period) x 100
= (500ns / 1µs) x 100
= 0.5 x 100
= 50%

Pulse delay

As shown in Figure 11-30, pulse delay is the time interval between the start of the rising pulse edge of the trigger output pulse and the output pulse. The polarity of the trigger output pulse edge to the output pulse can be configured (rising or falling). Pulse delay has a variable delay with respect to the trigger output.

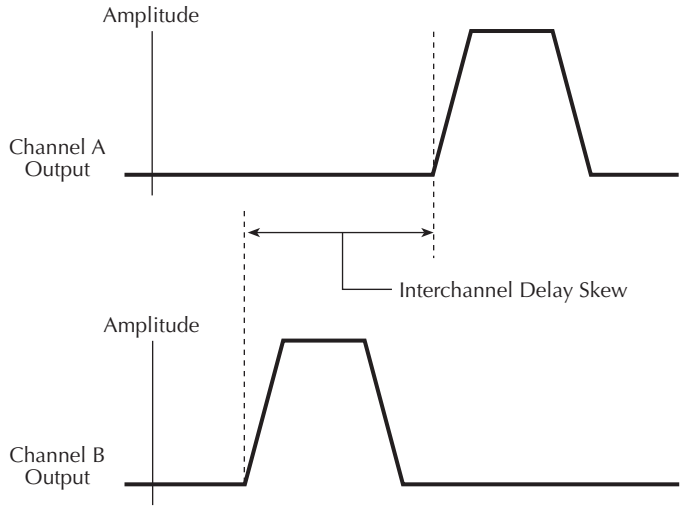
Figure 11-30
Pulse delay



Interchannel delay (skew)

As shown in Figure 11-31, interchannel delay is the time interval between the rising pulse edge of the two output channels (Channel A and Channel B). Skew can be adjusted through the use of the pulse delay for each individual channel.

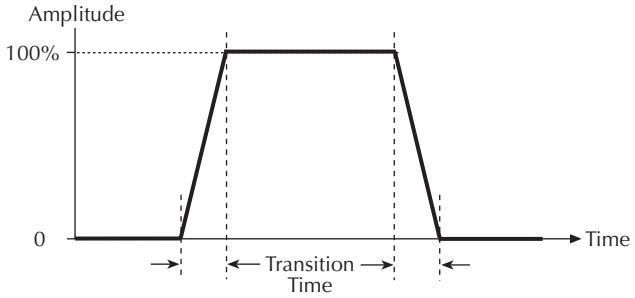
Figure 11-31
Interchannel delay (skew)



Transition time

As shown in Figure 11-32, transition time is the interval between corresponding 0% and 100% amplitude points on the rising/falling edge of the pulse.

Figure 11-32
Transition time

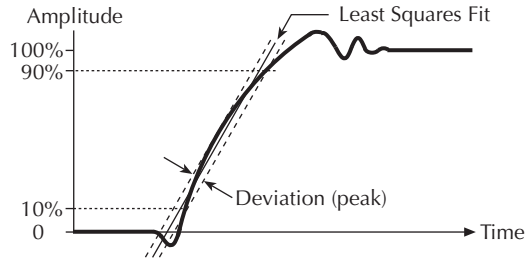


Linearity (deviation)

The slope for a linear pulse would be a least squares fit line for actual data between the 10% and 90% points of the transition. Figure 11-33 shows an example of this transition. The maximum deviation from the least squares fit line and the transition is expressed as a percentage of pulse amplitude.

For example, assume 100% amplitude is 1V and the maximum deviation is 70 mV. The deviation for this example is 7%.

Figure 11-33
Linearity (deviation)



Jitter

Jitter is the short-term instability of one edge relative to a reference edge. it is usually specified as an rms value, which is one standard deviation (or sigma). If distribution is assumed Gaussian, six sigma represents 99.74% of peak-to-peak jitter.

The reference edge for period jitter is the previous rising edge. The reference edge for delay jitter is the rising edge of the trigger output. Width jitter is the stability of the falling edge with respect to the rising edge.

Pulse levels

Pulse-level terms are shown in Figure 11-34. As shown, a pulse consists of a pulse base (low level) and a pulse top (high level). The low-to-high magnitude is the peak-to-peak amplitude of the pulse.

The pulse can be offset from zero volts. The low level will provide the offset for the pulse. For example, to achieve a pulse amplitude of 10 V peak-to-peak with a DC offset of 5 V, the high level voltage value would be set to 15 V and the low level voltage set to 5 V. Figure 11-35 shows the pulse levels for this example.

Figure 11-34
Pulse levels

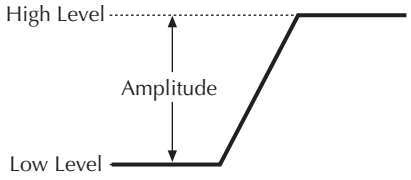
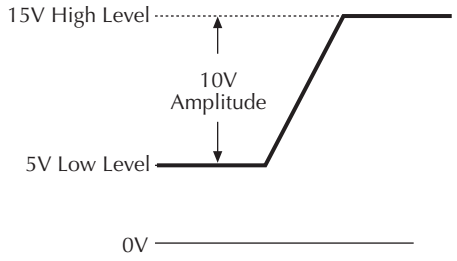


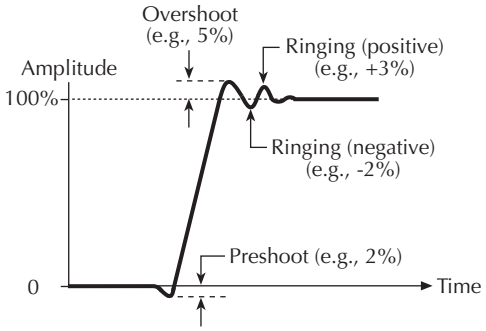
Figure 11-35
Pulse offset example



Distortion (preshoot, overshoot, and ringing)

Preshoot and overshoot are peak distortions preceding/following an edge. Ringing is the positive-peak and negative-peak distortion (excluding overshoot) on pulse top or base. Distortion for a pulse is shown in Figure 11-36. A combined preshoot, overshoot, and ringing specification of 5% implies an overshoot and undershoot <5% of pulse amplitude.

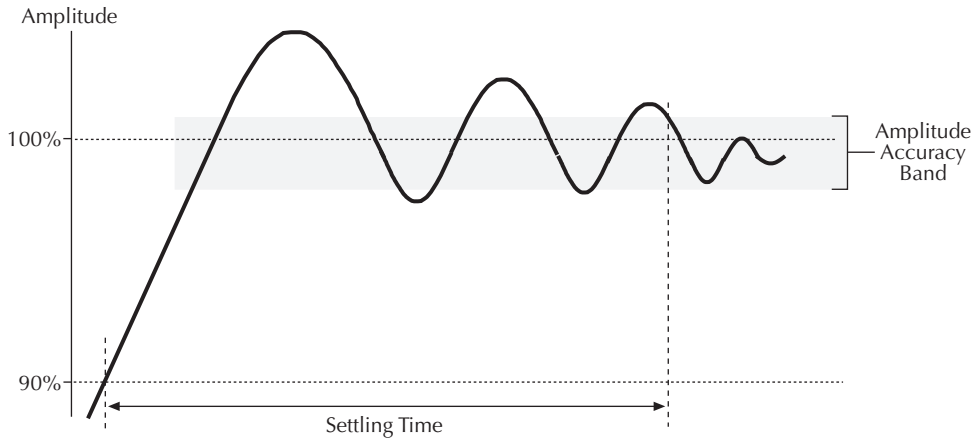
Figure 11-36
Preshoot, overshoot, and ringing



Settling time

As shown in Figure 11-37, settling time is the time it takes for pulse levels to settle within level specifications. Settling time is measured from the 90% point on the rising edge to the point where the pulse level remains in the accuracy band.

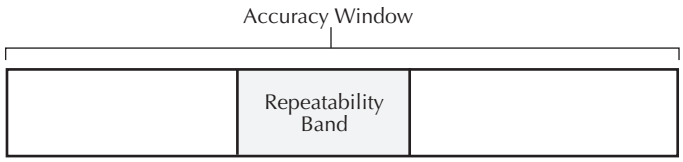
Figure 11-37
Settling time



Repeatability

When the pulse card operates under the same environmental conditions and with the same settings, the value of a parameter will lie within a band inside the accuracy window (see Figure 11-38). Repeatability defines the width of this band.

Figure 11-38
Repeatability



PIV-Q user libraries

The user library UTMs used in the PIV-Q package are documented as follows:

- "DualPulseulib UTM descriptions" on page 11-49 – The DualPulseulib UTMs are used to control the Dual Card pulse generator through the Model 4205-PCU (Pulse Combiner Unit). These UTMs are used in the Model 4200-PIV-Q package.
- "QPulseIVulib UTM descriptions" on page 11-61 – The QPulseIVulib UTMs are used for the Model 4200-PIV-Q package.

DualPulseulib UTM descriptions

The DualPulseulib UTMs are used to control the Dual Card pulse generator through the Model 4205-PCU (Pulse Combiner Unit). The Pulse Combiner Unit is supplied with the Model 4205-PIV-Q and provides dual pulse control of two adjacent Model 4205-PG2 pulse generators. For more information, refer to Section 12 of the Reference Manual and the PIVQ application in the Applications Manual.

The modules contained in the dual pulse user library are listed in Table 11-38 with detailed information following the table.

NOTE *Dual pulse control is by a usrlib, not within LPTLib. LPTLib commands such as devclr and devint do not apply to the Dual Card pulse generator. To initialize the Dual Card pulse generator, use the dpulse_init function.*

Table 11-38
DualPulseulib UTMs

User Module	Description
dpulse_burst_count	Sets the burst count for the Dual Card pulse generator.
dpulse_current_limit	Sets current limit for the Dual Card pulse generator.
dpulse_delay	Sets the pulse delay for the Dual Card pulse generator.
dpulse_fall	Sets the pulse fall time for the Dual Card pulse generator.
dpulse_float	Disconnects (floats) the output shield of Dual Card pulse generator from ground.
dpulse_halt	Stops all pulse output from the Dual Card pulse generator.
dpulse_init	Initializes the Dual Card pulse generator.
dpulse_load	Sets the output impedance of the Dual Card pulse generator.
dpulse_output	Turns the output of the Dual Card pulse generator on or off.
dpulse_output_mode	Sets the output mode for the Dual Card pulse generator.
dpulse_period	Sets the period for the Dual Card pulse generator.
dpulse_range	Sets the source range for the Dual Card pulse generator.
dpulse_rise	Sets the pulse rise time for the Dual Card pulse generator.
dpulse_trig	Initiates the start of pulse output, and sets the trigger mode.
dpulse_trig_polarity	Sets the trigger polarity of the Dual Card pulse generator.
dpulse_vhigh	Sets the high level value for pulse output of the Dual Card pulse generator.
dpulse_vlow	Sets the low level value for pulse output of the Dual Card pulse generator.
dpulse_width	Sets the pulse width for the Dual Card pulse generator.

dpulse_burst_count

Module Return Type: int
Number of Parameters: 1

Description This function sets the burst count for the Dual Card pulse generator. It sets the number of pulses (periods) that the pulse generator will output when used in the burst mode.

The parameter and return values for this function are listed in Table 11-39 and Table 11-40.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use dpulse_trig to set the trigger mode to burst.

Table 11-39
Parameter for pulse_burst_count

Name	Description
PulseCount	(long) The burst count for Dual Card pulse generator.

Table 11-40
Return values for dpulse_burst_count

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_current_limit

Module Return Type: int
Number of Parameters: 1

Description This function sets current limit for the Dual Card pulse generator. Maximum limit values are source range dependent:

10 V range (high speed): Maximum current limit is 400 mA.
40 V range (high voltage): Maximum current limit is 800 mA.

The current limit is used to protect the Device Under Test (DUT) by using the DUT load impedance setting to calculate the maximum voltage that will be output to the DUT.

The parameter and return values for this function are listed in Table 11-41 and Table 11-42.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use dpulse_load function to set the DUT load impedance.

Table 11-41
Parameter for dpulse_current_limit

Name	Description
PulseCurrentLimit	(double) The current limit of the pulse generator (amps).

Table 11-42
Return values for dpulse_current_limit

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_delay

Module Return Type: int
Number of Parameters: 1

Description This function sets the delay period from trigger to when the pulse output starts. Delay can be set from 0 s to (Period - 10 ns).

The parameter and return values for this function are listed in Table 11-43 and Table 11-44.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-43
Parameter for dpulse_delay

Name	Description
PulseDelay	(double) The pulse delay (seconds).

Table 11-44
Return values for dpulse_delay

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_fall

Module Return Type: int
Number of Parameters: 1

Description This function sets pulse fall time for the Dual Card pulse generator. Minimum fall time values are source range dependent:

10 V range (high speed): Minimum fall time is 10 ns.
40 V range (high voltage): Minimum fall time is 100 ns.

Based on system and interconnect bandwidth, the actual fall time may need to be greater than the minimum to obtain an acceptable pulse shape.

The parameter and return values for this function are listed in Table 11-45 and Table 11-46.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use the dpulse_range function to set the source range.

Table 11-45
Parameter for dpulse_fall

Name	Description
FallTime	(double) The fall time for the pulse output (seconds).

Table 11-46
Return values for dpulse_fall

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_float

Module Return Type: int
Number of Parameters: 1

Description This function floats (disconnects) the output shield of the Dual Card pulse generator from ground. The Floating setting is intended to be used in conjunction with a scope (sense output located on the Model 4205-PCU) and the PIVQ solution. The No Float setting connects the output shield to ground, bypassing the sense resistor through the Sense connector of the Model 4205-PCU.

The parameter and return values for this function are listed in Table 11-47 and Table 11-48.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-47
Parameter for dpulse_float

Name	Description
FloatEnable	(long) Sets the float of the pulse generator: 0 - No Float (pulse generator is grounded). 1 - Floating (pulse generator is not grounded and will need a ground through the Model 4200-PCU sense connection.

Table 11-48
Return values for dpulse_float

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_halt

Module Return Type: int
Number of Parameters: 0

Description This function stops all (halts) pulse output from the Dual Card pulse generator and turns the output off.

There are no parameters for this function. The return values are listed in Table 11-50.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use the dpulse_range function to set the source range.

Pulse output can be restarted by first turning the outputs back on using dpulse_output, and then starting pulse output using the dpulse_trig function.

Table 11-49
Parameter for dpulse_halt

Name	Description
(none)	N/A

Table 11-50
Return values for dpulse_halt

Value	Description
0	No Errors.

Table 11-50
Return values for dpulse_halt

Value	Description
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_init

Module Return Type: int
Number of Parameters: 2

Description This function initializes the Dual Card pulse generator. The Dual Card pulse generator consists of two adjacent Model 4205-PG2s that are electrically "stacked" on top of each other through the Model 4205-PCU (Pulse Combiner Unit). Initilization configures each card in the stack using the specified card IDs for the two pulse generator cards being stacked. Usually, the lower ID PG2 is the HighVpu. In the case where VPU1 and VPU2 are being stacked, VPU1 will be the HighVpu, and VPU2 will be the LowVpu.

The trigger output of the Model 4205-PCU is the same as the trigger output of the HighVpu. In addition to the output setup, this routine also configures the triggers.

The parameters and return values for this function are listed in Table 11-51 and Table 11-52.

Table 11-51
Parameters for dpulse_init

Name	Description
HighVpuId	The PG2 that is at the top of the stack (VPU1)
LowVpuId	The PG2 that is at the bottom of the stack (VPU2)

Table 11-52
Return values for dpulse_init

Value	Description
0	No Errors.
-1	LowVpuId is not in the system configuration.
-2	HighVpuId is not in the system configuration.
-3	HighVpuId and LowVpuId cannot be the same.
Other	All other error cards match up to LPT errors.

dpulse_load

Module Return Type: int
Number of Parameters: 1

Description This function sets the impedance of the load connected to the Dual Card pulse generator. This impedance setting determines the actual pulse voltages output by the pulse generator and is used for calculating the maximum voltage for the dpulse_current_limit function. Valid parameter values range from 1 Ω to 1 M Ω.

The parameter and return values for this function are listed in Table 11-53 and Table 11-54.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use the dpulse_range function to set the source range.

Table 11-53
Parameter for dpulse_load

Name	Description
PulseLoad	(double) The target (DUT) load impedance (ohms).

Table 11-54
Return values for dpulse_load

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_output

Module Return Type: int
Number of Parameters: 1

Description This function turns the output of the Dual Card pulse generator on or off.

The parameter and return values for this function are listed in Table 11-55 and Table 11-56.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function. With the pulse generator turned on, pulse output will start when a trigger is initiated by the dpulse_init function.

Table 11-55
Parameter for dpulse_output

Name	Description
State	(long) The state of the pulse generator output: 0 - Output is off. 1 - Output is on.

Table 11-56
Return values for dpulse_output

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_output_mode

Module Return Type: int
Number of Parameters: 1

Description This function sets the pulse output mode of the Dual Card pulse generator. There are two modes, NORMAL and COMPLEMENT. When in complement mode, the pulse generator's VHigh and VLow are swapped.

The parameter and return values for this function are listed in Table 11-57 and Table 11-58.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-57
Parameter for dpulse_output_mode

Name	Description
Mode	(long) The output mode for the pulse generator: 0 - Normal mode. 1 - Complement mode (VHigh and VLow are swapped).

Table 11-58
Return values for dpulse_output_mode

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_period

Module Return Type: int
Number of Parameters: 1

Description This function sets the period for the Dual Card pulse generator. The maximum period is one second, while the minimum period depends on the rise time, fall time, and pulse width. The minimum period is calculated as follows:

Minimum Period = (Rise Time / 2) + Pulse Width + (Fall Time / 2) + 10 ns

The parameter and return values for this function are listed in Table 11-59 and Table 11-60.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-59
Parameter for dpulse_period

Name	Description
Period	(Double) The period for pulse output (seconds).

Table 11-60
Return values for dpulse_period

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_range

Module Return Type: int
Number of Parameters: 1

Description This function sets the source range for the Dual Card pulse generator. There are two source ranges: High Speed (±10 V into 50 Ω load) and High Voltage (±40 V into 50 Ω load).

The parameter and return values for this function are listed in Table 11-61 and Table 11-62.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-61
Parameter for dpulse_range

Name	Description
Range	(double) The range for the pulse generator: ±10 V - High Speed. ±40 V - High Voltage.

Table 11-62
Return values for dpulse_range

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_rise

Module Return Type: int
Number of Parameters: 1

Description This function sets pulse rise time for the Dual Card pulse generator. Minimum rise time values are source range dependent:

10 V range (high speed): Minimum rise time is 10 ns.
40 V range (high voltage): Minimum rise time is 100 ns.

Based on system and interconnect bandwidth, the actual rise time may need to be greater than the minimum to obtain an acceptable pulse shape.

The parameter and return values for this function are listed in Table 11-63 and Table 11-64.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function, and use the dpulse_range function to set the source range.

Table 11-63
Parameter for dpulse_rise

Name	Description
RiseTime	(double) The rise time for the pulse output (seconds).

Table 11-64
Return values for dpulse_rise

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_trig

Module Return Type: int
Number of Parameters: 1

Description This function is used to initiate the start of pulse output on the Dual Card pulse generator. It also selects the trigger mode: burst or continuous.

The parameter and return values for this function are listed in Table 11-65 and Table 11-66.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function. If using the Burst Mode, set the burst count with the dpulse_burst_count function.

Table 11-65
Parameter for dpulse_trig

Name	Description
Mode	(long) The pulse output mode: 0 - Burst Mode (using the dpulse_burst_count). 1 - Continuous Mode.

Table 11-66
Return values for dpulse_trig

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_trig_polarity

Module Return Type: int
Number of Parameters: 1

Description This function sets the trigger polarity of the Dual Card pulse generator output trigger. The trigger is a TTL level output at the same frequency (period) as the pulse output.

The parameter and return values for this function are listed in Table 11-67 and Table 11-68.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-67
Parameter for dpulse_trig_polarity

Name	Description
TrigPolarity	(long) The trigger polarity: 0 - Negative trigger polarity. 1 - Positive trigger polarity.

Table 11-68
Return values for dpulse_trig_polarity

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_vhigh

Module Return Type: int
Number of Parameters: 1

Description This function sets the high voltage value of the output pulse of the Dual Card pulse generator. The output pulse has two levels, vlow and vhigh. The level is dependent on the source range and pulse load. The difference between pulse low and pulse high is the peak-to-peak amplitude of the output pulse.

50 Ω Load:

10 V Range (High Speed) – Pulse high and pulse low can be set between -10 V and +10 V.

40 V Range (High Voltage) – Pulse high and pulse low can be set between -40 V and +40 V.

1 M Ω Load:

10 V Range (High Speed) – Pulse high and pulse low can be set between -20 V and +20 V.

40 V Range (High Voltage) – Pulse high and pulse low can be set between -80 V and +80 V.

Related functions: dpulse_current_limit, dpulse_range, dpulse_vlow

The parameter and return values for this function are listed in Table 11-69 and Table 11-70.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-69
Parameter for dpulse_vhigh

Name	Description
PulseVHigh	(double) The pulse high value (V).

Table 11-70
Return values for dpulse_vhigh

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_vlow

Module Return Type: int
Number of Parameters: 1

Description This function sets the low voltage value of the output pulse of the Dual Card pulse generator. The output pulse has two levels, vlow and vhigh. The level is dependent on the source range and pulse load. The difference between pulse low and pulse high is the peak-to-peak amplitude of the output pulse.

50 Ω Load:

10 V Range (High Speed) – Pulse high and pulse low can be set between -10 V and +10 V.

40 V Range (High Voltage) – Pulse high and pulse low can be set between -40 V and +40 V.

1 M Ω Load:

10 V Range (High Speed) – Pulse high and pulse low can be set between -20 V and +20 V.

40 V Range (High Voltage) – Pulse high and pulse low can be set between -80 V and +80 V.

Related functions: dpulse_current_limit, dpulse_range, dpulse_vhigh

The parameter and return values for this function are listed in Table 11-41 and Table 11-42.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-71
Parameter for dpulse_vlow

Name	Description
PulseVLow	(double) The pulse low value (V).

Table 11-72
Return values for dpulse_vlow

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

dpulse_width

Module Return Type: int
Number of Parameters: 1

Description This function sets the pulse width of the Dual Card pulse generator. The pulse width includes half of the rise time and half of the fall time (FWHM: full width @ half maximum). The maximum width depends on the range:

10 V range (high speed): 10 ns to (period - 10 ns)
40 V range (high voltage): 100 ns to (period - 100 ns)

Based on system and interconnect bandwidth, the actual pulse width may need to be greater than the minimum to obtain an acceptable pulse shape.

The parameter and return values for this function are listed in Table 11-73 and Table 11-74.

Procedure Before using this function, make sure the Dual Card pulse generator has been initialized using the dpulse_init function.

Table 11-73
Parameter for dpulse_width

Name	Description
PulseWidth	(double) The pulse width for the Dual Card pulse generator.

Table 11-74
Return values for dpulse_width

Value	Description
0	No Errors.
-1	Dual Card pulse generator has not been initialized.
Other	All other error cards match up to LPT errors.

QPulseIVulib UTM descriptions

The UTMs used for the Model 4200-PIV-Q package are summarized in Table 11-75. Details for the UTMs follow the table.

Table 11-75
QPulseIVulib UTMs

User Module	Description
CableCompensation_QPulseIV	Used to determine the compensation factors required to account for cable losses incurred during the pulse and DC sweeps for the Model 4200-PIV-Q package.
ScopeShot_FET_QPulseIV	Used to run a single pulse/point measurement on a FET DUT (FET, HEMT, and so on.) using the Model 4200-PIV-Q package.
Vd_Id_Pulse_DC_Family_QPulseIV	Used to perform a Pulsed vs. DC Vd-Id sweep using the Model 4200-PIV-Q package.
Vd_Id_Single_DC_QPulseIV	Used to perform a single DC Vd-Id sweep using the Model 4200-PIV-Q package.
Vd_Id_Single_Pulse_QPulseIV	Used to perform a single pulsed Vd-Id sweep using the Model 4200-PIV-Q package.
Vg_Id_Pulse_DC_QPulseIV	Used to perform a Pulsed vs. DC Vg-Id sweep using the Model 4200-PIV-Q package.
Vg_Id_Single_DC_QPulseIV	Used to perform a DC Vg-Id sweep using the Model 4200-PIV-Q package.
Vg_Id_Single_Pulse_QPulseIV	Used to perform a Pulsed Vg-Id sweep using the Model 4200-PIV-Q package.

Return codes:

All modules share the following return codes:

0	No Errors
-15001	PIV-Q hardware has not been initialized
-15002	Unable to trigger the Gate/Base pulse generator
-15003	Unable to trigger the Drain/Collector pulse generator
-15004	Unable to obtain a scope reading
-15005	Unable to set scope cursors
-15006	Unable to use load line, as the required voltage is beyond source range
-15007	Invalid Gate/Base PG2 ID Specified
-15008	Invalid Drain/Collector PG2 High ID Specified
-15009	Invalid Drain/Collector PG2 Low ID Specified
-15010	Invalid Gate/Base SMU ID Specified
-15011	Invalid Drain/Collector SMU ID Specified
-15012	Unable to force given voltage on Gate/Base
-15013	Unable to force given voltage on Drain/Collector
-15014	Invalid Cable Compensation Type (not OPEN or THRU)
-15015	Unable to find Cable Compensation factor in file
-15016	Unable to write Cable Compensation to file
-15017	Unable to allocate enough memory for test
-15018	Unable to close/open the Gate/Base SMU relays
-15019	Unable to close/open the Drain SMU relays
-15020	Unable to close/open the Gate/Base PG2 Solid State Relays (HEOR)
-15021	Unable to close/open the Drain/Collector Solid State Relays (HEOR)
-15022	Unable to initialize the scope
-15023	Unable to set scope impedance
-15024	Unable to set the scope range
-15025	Unable to configure the scope measure
-15026	Unable to set the scope sample rate
-15027	Unable to set the scope offset
-15028	Unable to set the scope to external triggering
-15029	Unable to set the average number desired on the scope
-15030	Unable to set the scope trigger delay
-15031	Unable to read waveform from scope
-15032	Unable to set the scope delay
-15033	Unable to capture waveform on scope
-15034	Unable to find Gate/Base PG2 calibration values
-15035	Unable to find Drain/Collector High PG2 calibration values
-15036	Unable to find Drain/Collector Low PG2 calibration values
-15037	Unable to initialize Gate/Base PG2
-15038	Unable to initialize Drain/Collector High PG2
-15039	Unable to initialize Drain/Collector Low PG2
-15040	Unable to initialize the Model 4205-PCU
-15041	Unable to set the range on the Model 4205-PCU

-15042 Unable to set the period on the Model 4205-PCU
-15043 Unable to set rise time on the Model 4205-PCU
-15044 Unable to set fall time on the Model 4205-PCU
-15045 Unable to set pulse width on the Model 4205-PCU
-15046 Unable to set the Qpoint on the Model 4205-PCU
-15047 Unable to set the load on the Model 4205-PCU
-15048 Unable to set the burst count on the Model 4205-PCU
-15049 Unable to set the delay on the Model 4205-PCU
-15050 Unable to set the current limit on the Model 4205-PCU
-15051 Unable to turn on/off the output on the Model 4205-PCU
-15052 Unable to float the Model 4205-PCU
-15053 Unable to set VHigh on the Model 4205-PCU
-15054 Unable to set VLow on the Model 4205-PCU
-15055 Unable to trigger the Model 4205-PCU
-15056 Unable to halt the Model 4205-PCU
-15057 Unable to initialize the Gate/Base PG2
-15058 Unable to set the range on the Gate/Base PG2
-15059 Unable to set the period on the Gate/Base PG2
-15060 Unable to set the rise time on the Gate/Base PG2
-15061 Unable to set the fall time on the Gate/Base PG2
-15062 Unable to set the pulse width on the Gate/Base PG2
-15063 Unable to set the QPoint on the Gate/Base PG2
-15064 Unable to set the burst count on the Gate/Base PG2
-15065 Unable to set the trigger source on the Gate/Base PG2
-15066 Unable to set the current limit on the Gate/Base PG2
-15067 Unable to set the load on the Gate/Base PG2
-15068 Unable to turn the output of the Gate/Base PG2 on/off
-15069 Unable to halt the Gate/Base PG2

CableCompensation_QPulseIV

Module Return Type: int
Number of Parameters: 5

Arguments:

GateSMU,char *,Input,SMU1,,
DrainSMU,char *,Input,SMU2,,
Getlevel,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,

Description:

The CableCompensation_QPulseIV function is used to determine the compensation factors required to account for cable losses incurred during the pulse and DC sweeps in the Model 4200-PIV-Q package. This compensation is unique to this package and does not apply to any other configurations. Please refer to the manual for proper PIV-Q setup and interconnect instructions.

There are two parts to this compensation. The first part is an OPEN test that will take a few seconds to complete. The second part is a THRU/SHORT that will take about two minutes to complete. The THRU/SHORT test requires a THRU structure (on wafer) or some method to connect the Gate/Base and Drain/Collector signals together. Both parts are required for a complete compensation and it is recommended that a compensation be run every time the setup has been moved or has changed in any way. Before running this routine perform an Autocal on the scope card.

The Model 4205-PCU combines two Model 4205-PG2 cards into a single, higher power pulse channel to provide power to the DUT Drain/Collector.

Inputs:

GateSMU	(char*) String representing the ID for the SMU connected to the gate (for example, SMU1).
DrainSMU	(char*) String representing the ID for the SMU connected to the drain (for example, SMU2).
GateVPU	(char*) String representing the ID for the PG2 connected to the gate (for example, VPU1).
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system (for example, VPU2). DrainVPULow (char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system (for example, VPU3).

Outputs:

NONE

ScopeShot_FET_QPulseIV

Module Return Type: int
Number of Parameters: 34

Arguments:

Vgs,double,Input,2,,
VgQPoint,double,Input,0,,
Vds,double,Input,4,,
VdQPoint,double,Input,0,,
PulseWidth,double,Input,300e-9,,
PulsePeriod,double,Input,1e-6,,
RiseTime,double,Input,100e-9,,
FallTime,double,Input,100e-9,,
Average,int,Input,100,,
GateVPURange,double,Input,5,,
DrainVPURange,double,Input,5,,
GateScpRange,double,Input,0,,
DrainScpRange,double,Input,0,,
GateCompliance,double,Input,.1,,
DrainCompliance,double,Input,.1,,
GateLoadLine,int,Input,1,,
DrainLoadLine,int,Input,1,,

```
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
Time,D_ARRAY_T,Output,,,
TimeSize,int,Input,10000,,
Ch1Out,D_ARRAY_T,Output,,,
Ch1OutSize,int,Input,10000,,
Ch2Out,D_ARRAY_T,Output,,,
Ch2OutSize,int,Input,10000,,
GateV,double *,Output,,,
GateI,double *,Output,,,
DrainV,double *,Output,,,
DrainI,double *,Output,,,
MeasurementCursor1,double *,Output,,,
MeasurementCursor2,double *,Output,,,
```

Description:

The ScopeShot_FET_QPulseIV function is used to run a single pulse/point measurement on a FET DUT (FET, HEMT, and so on.) using the Model 4200-PIV-Q package. This measurement is unique to this package and should not be used with any other setups, as unexpected signals may be provided and/or improper results may be reported. Please refer to the manual for proper PIV-Q setup instructions.

ScopeShot will only perform pulse measurements based on a single set of parameters (no sweeping). Also, this routine will source more than the specified number of pulses in Average if either GateScpRange or DrainScpRange are auto-range, or GateLoadLine or DrainLoadLine are = 1.

All voltage levels specified below assume a 50 Ω DUT load.

Inputs:

Vgs	(double) The pulsed Gate-Source voltage bias. Vgs is range dependent and can be between -5 V to +5 V or between -20 V to +20 V.
VgQPoint	(double) The base, or bias point, of the pulsed Gate source. VgQPoint is range dependent and can be between -5 V to +5 V or between -20 V to +20 V.
Vds	(double) The pulsed Drain-Source voltage bias. Vds is range dependent and can be between -10 V to +10 V or between -40 V to +40 V.
VdQPoint	(double) The base, or bias point, of the pulsed Gate source. VdQPoint is range dependent and can be between -10 V to +10 V or between -40 V to +40 V.
PulseWidth	(double) The Vgs and Vds pulse width (PW). The Pulse Width can be 300 ns to (1s - 10 ns) in 10 ns resolution steps.
PulsePeriod	(double) The pulse period for both Vgs and Vds. Minimum period is (FallTime/2) + (RiseTime/2) + PulseWidth + 10 ns. Maximum period is 1s.
RiseTime	(double) The transition time from the Qpoint to the pulse value for both Vgs and Vds.The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the

	minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
FallTime	(double) The transition time from the pulse value to the Qpoint value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
Average	(int) The number of pulses to average (recommend 100 for a reasonable result, use larger for lower current measurements).
GateVPURange	(double) The source range for DUT Gate PG2. Valid ranges are 5 V (High Speed) and 20 V (High Voltage).
DrainVPURange	(double) The source range for the DUT Drain Model 4205-PCU. Valid ranges are 10 V (High Speed) and 40 V (High Voltage).
GateScpRange	(double) The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
DrainScpRange	(double) The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
GateCompliance	(double) The current compliance on the gate. This supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
DrainCompliance	(double) The current compliance on the drain. This supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
GateLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Gate. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
DrainLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
GateSMU	(char*) String representing the ID for the SMU connected to the gate.
DrainSMU	(char*) String representing the ID for the SMU connected to the drain.
GateVPU	(char*) String representing the ID for the PG2 connected to the gate.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.

DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
TimeSize Ch1OutSize Ch2OutSize	(int) Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.

Outputs:

Time	(double) Array of time values corresponding to the pulsed scope shot.
Ch1Out	(double) Array of Channel 1 (Gate) values as seen on the scope.
Ch2Out	(double) Array of Channel 2 (Drain) values as seen on the scope.
GateV	(double) Measured Gate Voltage. GateI(double) Measured Gate Current.
DrainV	(double) Measured Drain Voltage. DrainI(double) Measured Drain Current.
MeasurementCursor1	(double) Cursor used to measure the voltage and current values on the scope.
MeasurementCursor2	(double) Cursor used to measure the voltage and current values on the scope.

Vd_Id_Pulse_DC_Family_QPulseIV

Module Return Type: int
Number of Parameters: 195

Arguments:

VgStart,double,Input,1.5,,
VgStop,double,Input,2.5,,
VgNumSteps,int,Input,3,,
VdStart,double,Input,0,,
VdStop,double,Input,4,,
VdStep,double,Input,.05,,
VgQPoint,double,Input,0,,
VdQPoint,double,Input,0,,
PulseWidth,double,Input,300e-9,,
PulsePeriod,double,Input,1e-6,,
RiseTime,double,Input,100e-9,,
FallTime,double,Input,100e-9,,
AverageNum,int,Input,100,,
GateVPURange,double,Input,5,,
DrainVPURange,double,Input,5,,
GateSMURange,int,Input,1,,
DrainSMURange,int,Input,1,,
GateScpRange,double,Input,0,,
DrainScpRange,double,Input,0,,
GateLoadLineCorrection,int,Input,1,,
DrainLoadLineCorrection,int,Input,1,,
GateCompliance,double,Input,.1,,
DrainCompliance,double,Input,.1,,

```
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
NPLC,double,Input,.01,,
DCSourceDelay,double,Input,0,,
DC_vs_Pulse,int,Input,2,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV_DC_1,D_ARRAY_T,Output,,,
DrainV_DC_1_Size,int,Input,10000,,
DrainI_DC_1,D_ARRAY_T,Output,,,
DrainI_DC_1_Size,int,Input,10000,,
GateV_DC_1,D_ARRAY_T,Output,,,
GateV_DC_1_Size,int,Input,10000,,
GateI_DC_1,D_ARRAY_T,Output,,,
GateI_DC_1_Size,int,Input,10000,,
DrainV_DC_2,D_ARRAY_T,Output,,,
DrainV_DC_2_Size,int,Input,10000,,
DrainI_DC_2,D_ARRAY_T,Output,,,
DrainI_DC_2_Size,int,Input,10000,,
GateV_DC_2,D_ARRAY_T,Output,,,
GateV_DC_2_Size,int,Input,10000,,
GateI_DC_2,D_ARRAY_T,Output,,,
GateI_DC_2_Size,int,Input,10000,,
DrainV_DC_3,D_ARRAY_T,Output,,,
DrainV_DC_3_Size,int,Input,10000,,
DrainI_DC_3,D_ARRAY_T,Output,,,
DrainI_DC_3_Size,int,Input,10000,,
GateV_DC_3,D_ARRAY_T,Output,,,
GateV_DC_3_Size,int,Input,10000,,
GateI_DC_3,D_ARRAY_T,Output,,,
GateI_DC_3_Size,int,Input,10000,,
DrainV_DC_4,D_ARRAY_T,Output,,,
DrainV_DC_4_Size,int,Input,10000,,
DrainI_DC_4,D_ARRAY_T,Output,,,
DrainI_DC_4_Size,int,Input,10000,,
GateV_DC_4,D_ARRAY_T,Output,,,
GateV_DC_4_Size,int,Input,10000,,
GateI_DC_4,D_ARRAY_T,Output,,,
GateI_DC_4_Size,int,Input,10000,,
```

```
DrainV_DC_5,D_ARRAY_T,Output,,,
DrainV_DC_5_Size,int,Input,10000,,
DrainI_DC_5,D_ARRAY_T,Output,,,
DrainI_DC_5_Size,int,Input,10000,,
GateV_DC_5,D_ARRAY_T,Output,,,
GateV_DC_5_Size,int,Input,10000,,
GateI_DC_5,D_ARRAY_T,Output,,,
GateI_DC_5_Size,int,Input,10000,,
DrainV_DC_6,D_ARRAY_T,Output,,,
DrainV_DC_6_Size,int,Input,10000,,
DrainI_DC_6,D_ARRAY_T,Output,,,
DrainI_DC_6_Size,int,Input,10000,,
GateV_DC_6,D_ARRAY_T,Output,,,
GateV_DC_6_Size,int,Input,10000,,
GateI_DC_6,D_ARRAY_T,Output,,,
GateI_DC_6_Size,int,Input,10000,,
DrainV_DC_7,D_ARRAY_T,Output,,,
DrainV_DC_7_Size,int,Input,10000,,
DrainI_DC_7,D_ARRAY_T,Output,,,
DrainI_DC_7_Size,int,Input,10000,,
GateV_DC_7,D_ARRAY_T,Output,,,
GateV_DC_7_Size,int,Input,10000,,
GateI_DC_7,D_ARRAY_T,Output,,,
GateI_DC_7_Size,int,Input,10000,,
DrainV_DC_8,D_ARRAY_T,Output,,,
DrainV_DC_8_Size,int,Input,10000,,
DrainI_DC_8,D_ARRAY_T,Output,,,
DrainI_DC_8_Size,int,Input,10000,,
GateV_DC_8,D_ARRAY_T,Output,,,
GateV_DC_8_Size,int,Input,10000,,
GateI_DC_8,D_ARRAY_T,Output,,,
GateI_DC_8_Size,int,Input,10000,,
DrainV_DC_9,D_ARRAY_T,Output,,,
DrainV_DC_9_Size,int,Input,10000,,
DrainI_DC_9,D_ARRAY_T,Output,,,
DrainI_DC_9_Size,int,Input,10000,,
GateV_DC_9,D_ARRAY_T,Output,,,
GateV_DC_9_Size,int,Input,10000,,
GateI_DC_9,D_ARRAY_T,Output,,,
GateI_DC_9_Size,int,Input,10000,,
DrainV_DC_10,D_ARRAY_T,Output,,,
DrainV_DC_10_Size,int,Input,10000,,
DrainI_DC_10,D_ARRAY_T,Output,,,
DrainI_DC_10_Size,int,Input,10000,,
```

```
GateV_DC_10,D_ARRAY_T,Output,,,
GateV_DC_10_Size,int,Input,10000,,
GateI_DC_10,D_ARRAY_T,Output,,,
GateI_DC_10_Size,int,Input,10000,,
DrainV_Pulse_1,D_ARRAY_T,Output,,,
DrainV_Pulse_1_Size,int,Input,10000,,
DrainI_Pulse_1,D_ARRAY_T,Output,,,
DrainI_Pulse_1_Size,int,Input,10000,,
GateV_Pulse_1,D_ARRAY_T,Output,,,
GateV_Pulse_1_Size,int,Input,10000,,
GateI_Pulse_1,D_ARRAY_T,Output,,,
GateI_Pulse_1_Size,int,Input,10000,,
DrainV_Pulse_2,D_ARRAY_T,Output,,,
DrainV_Pulse_2_Size,int,Input,10000,,
DrainI_Pulse_2,D_ARRAY_T,Output,,,
DrainI_Pulse_2_Size,int,Input,10000,,
GateV_Pulse_2,D_ARRAY_T,Output,,,
GateV_Pulse_2_Size,int,Input,10000,,
GateI_Pulse_2,D_ARRAY_T,Output,,,
GateI_Pulse_2_Size,int,Input,10000,,
DrainV_Pulse_3,D_ARRAY_T,Output,,,
DrainV_Pulse_3_Size,int,Input,10000,,
DrainI_Pulse_3,D_ARRAY_T,Output,,,
DrainI_Pulse_3_Size,int,Input,10000,,
GateV_Pulse_3,D_ARRAY_T,Output,,,
GateV_Pulse_3_Size,int,Input,10000,,
GateI_Pulse_3,D_ARRAY_T,Output,,,
GateI_Pulse_3_Size,int,Input,10000,,
DrainV_Pulse_4,D_ARRAY_T,Output,,,
DrainV_Pulse_4_Size,int,Input,10000,,
DrainI_Pulse_4,D_ARRAY_T,Output,,,
DrainI_Pulse_4_Size,int,Input,10000,,
GateV_Pulse_4,D_ARRAY_T,Output,,,
GateV_Pulse_4_Size,int,Input,10000,,
GateI_Pulse_4,D_ARRAY_T,Output,,,
GateI_Pulse_4_Size,int,Input,10000,,
DrainV_Pulse_5,D_ARRAY_T,Output,,,
DrainV_Pulse_5_Size,int,Input,10000,,
DrainI_Pulse_5,D_ARRAY_T,Output,,,
DrainI_Pulse_5_Size,int,Input,10000,,
GateV_Pulse_5,D_ARRAY_T,Output,,,
GateV_Pulse_5_Size,int,Input,10000,,
GateI_Pulse_5,D_ARRAY_T,Output,,,
GateI_Pulse_5_Size,int,Input,10000,,
```

```
DrainV_Pulse_6,D_ARRAY_T,Output,,,
DrainV_Pulse_6_Size,int,Input,10000,,
DrainI_Pulse_6,D_ARRAY_T,Output,,,
DrainI_Pulse_6_Size,int,Input,10000,,
GateV_Pulse_6,D_ARRAY_T,Output,,,
GateV_Pulse_6_Size,int,Input,10000,,
GateI_Pulse_6,D_ARRAY_T,Output,,,
GateI_Pulse_6_Size,int,Input,10000,,
DrainV_Pulse_7,D_ARRAY_T,Output,,,
DrainV_Pulse_7_Size,int,Input,10000,,
DrainI_Pulse_7,D_ARRAY_T,Output,,,
DrainI_Pulse_7_Size,int,Input,10000,,
GateV_Pulse_7,D_ARRAY_T,Output,,,
GateV_Pulse_7_Size,int,Input,10000,,
GateI_Pulse_7,D_ARRAY_T,Output,,,
GateI_Pulse_7_Size,int,Input,10000,,
DrainV_Pulse_8,D_ARRAY_T,Output,,,
DrainV_Pulse_8_Size,int,Input,10000,,
DrainI_Pulse_8,D_ARRAY_T,Output,,,
DrainI_Pulse_8_Size,int,Input,10000,,
GateV_Pulse_8,D_ARRAY_T,Output,,,
GateV_Pulse_8_Size,int,Input,10000,,
GateI_Pulse_8,D_ARRAY_T,Output,,,
GateI_Pulse_8_Size,int,Input,10000,,
DrainV_Pulse_9,D_ARRAY_T,Output,,,
DrainV_Pulse_9_Size,int,Input,10000,,
DrainI_Pulse_9,D_ARRAY_T,Output,,,
DrainI_Pulse_9_Size,int,Input,10000,,
GateV_Pulse_9,D_ARRAY_T,Output,,,
GateV_Pulse_9_Size,int,Input,10000,,
GateI_Pulse_9,D_ARRAY_T,Output,,,
GateI_Pulse_9_Size,int,Input,10000,,
DrainV_Pulse_10,D_ARRAY_T,Output,,,
DrainV_Pulse_10_Size,int,Input,10000,,
DrainI_Pulse_10,D_ARRAY_T,Output,,,
DrainI_Pulse_10_Size,int,Input,10000,,
GateV_Pulse_10,D_ARRAY_T,Output,,,
GateV_Pulse_10_Size,int,Input,10000,,
GateI_Pulse_10,D_ARRAY_T,Output,,,
GateI_Pulse_10_Size,int,Input,10000,,
```

Description:

The Vd_Id_Pulse_DC_Family_QPulseIV sweep is used to perform a Pulsed vs. DC Vd-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical Vd-Id but only two sources

are used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2 channel scope, Model 4200-SCP2HR.

To create a family of curves, choose an appropriate start and stop value for Vgs, and a number of steps. This routine can run the sweeps in three different ways: 1) DC only; 2) Pulse only; 3) Pulse and DC curves. This routine supports from 1 to 10 Vd-Id curves based on up to ten different Vgs values.

All voltage levels specified below assume a 50 Ω DUT load.

Inputs:

VgStart	(double) The starting step value for Vg. For DC only sweeps, VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V or -20 V to +20 V depending on the specified source range for the PG2.
VgStop	(double) The final step value for Vg. For DC only sweeps, VgStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V or -20 V to +20 V depending on the specified source range for the PG2.
VgNumSteps	(double) The number of steps for Vg (Max = 10).
VdStart	(double) The starting sweep value for Vd. For DC only sweeps, VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VdStart must be between -10 V to +10 V or -40 V to +40 V depending on the specified source range for the Model 4205-PCU.
VdStop	(double) The final sweep value for Vd. For DC only sweeps, VdStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -10 V to +10 V or -40 V to +40 V depending on the specified source range for the Model 4205-PCU.
VdStep	(double) The number of steps for the Vd sweep. (Max = 10000)
VgQPoint	(double) The base value, or bias point, of the pulsed gate source. VgQPoint is range dependent and must be between -5 V to +5 V or -20 V to +20 V.
VdQPoint	(double) The base value, or bias point, of the pulsed drain sweep. VdQPoint is range dependent and must be between -10 V to +10 V or -40 V to +40 V.
PulseWidth	(double) The Vgs and Vds pulse width (PW). The Pulse Width range: 300 ns to (1s - 10 ns) in 10 ns resolution steps.
PulsePeriod	(double) The pulse period for Vgs and Vds. Minimum period is (FallTime/2) + (RiseTime/2) + PulseWidth + 10 ns. Maximum period is 1s.
RiseTime	(double) The transition time from the Qpoint to the pulse value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
FallTime	(double) The transition time from the pulse to the Qpoint value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.

Average	(int) The number of pulses to average (recommend 100 for a reasonable result, use larger for lower current measurements).
GateVPURange	(double) The source range for gate side PG2. Valid ranges are 5 V (High Speed) and 20 V (High Voltage).
DrainVPURange	(double) The source range for the drain side Model 4205-PCU. Valid ranges are 10 V (High Speed) and 40 V (High Voltage).
GateSMURange	(int) The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents.: 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
DrainSMURange	(int) The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents: 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
GateScpRange	(double) The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
DrainScpRange	(double) The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
GateLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Gate. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values.

	When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
DrainLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
GateCompliance	(double) The current compliance for the DUT Gate. This supplied value is used for the maximum source current for both DC and pulse source modes. For pulse current compliance, the supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
DrainCompliance	(double) The current compliance for DUT Drain. This supplied value is used for the maximum source current for both DC and pulse source modes. For pulse current compliance, the supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
MaxIg	(double) The max current allowed for the DUT Gate, for both DC and pulse sweeps. If the Gate current becomes greater than MaxIg, the present sweep will stop and the test will continue with the next Vg step.
MaxId	(double) The max current allowed on the DUT Drain, for both DC and pulse sweeps. If the Drain current becomes greater than MaxId, the present sweep will stop and the test will continue with the next Vg step.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the Gate power becomes greater than the MaxPowerGate, the present sweep will stop and the test will continue with the next Vg step.
MaxPowerDrain	(double) The max power allowed to the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the present sweep will stop and the test will continue with the next Vg step.
NPLC	(double) The DC measurement integration time in NPLC (Number of Power Line cycles).
DCSweepDelay	(double) Time, in seconds, between the DC source and measure for each sweep point.
DC_vs_Pulse	(int) Determines whether to run a DC and Pulse test or a DC only test or a Pulse only test.0 - Pulse Only, 1 - DC Only, 2 - DC and Pulse.
GateSMU	(char*) String representing the ID for the SMU connected to the DUT Gate.
DrainSMU	(char*) String representing the ID for the SMU connected to the DUT Drain.
GateVPU	(char*) String representing the ID for the PG2 connected to the DUT Gate.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU for the DUT Drain. This is the card on the right (or the card with the lower ID number), when facing the back of the system.
DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU for the DUT Drain. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
DrainV_DC_X_Size DrainI_DC_X_Size GateV_DC_X_Size GateI_DC_X_Size DrainV_Pulse_X_Size DrainI_Pulse_X_Size	(int) Sizes of the output arrays. All arrays should be the same size, and need to be large enough to hold all sweep points.

GateV_Pulse_X_Size
GateI_Pulse_X_Size

Outputs:

DrainV_DC_X
DrainV_Pulse_X (double) Array of programmed drain voltage values.

DrainI_DC_X
DrainI_Pulse_X (double) Array of measured drain currents.

GateV_DC_X
GateV_Pulse_X (double) Array of measured gate voltages.

GateI_DC_X
GateI_Pulse_X (double) Array of measured gate currents.

Vd_Id_Single_DC_QPulseIV

Module Return Type: int
Number of Parameters: 28

Arguments:

Vgs,double,Input,2,,
VdStart,double,Input,0,,
VdStop,double,Input,4,,
VdStep,double,Input,.05,,
GateRange,int,Input,1,,
GateCompliance,double,Input,.1,,
DrainRange,int,Input,1,,
DrainCompliance,double,Input,.1,,
NPLC,double,Input,.01,,
SweepDelay,double,Input,0,,
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV,D_ARRAY_T,Output,,,
DrainVSize,int,Input,10000,,
DrainI,D_ARRAY_T,Output,,,
DrainISize,int,Input,10000,,
GateV,D_ARRAY_T,Output,,,
GateVSize,int,Input,10000,,
GateI,D_ARRAY_T,Output,,,
GateISize,int,Input,10000,,
PostData,int,Input,1,,

Description:

The Vd_Id_Single_DC_QPulseIV sweep is used to perform a single DC Vd-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical DC Vd-Id but only two DC sources are used: one for the DUT Gate and one for the DUT Drain.

To create a family of curves, either change Vgs and run the test using append or use the Vd_Id_Pulse_DC_Family_QPulseIV function.

Inputs:

Vgs	(double) The DC gate-source voltage bias. Vgs must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VdStart	(double) The starting sweep value for Vd. VdStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VdStop	(double) The final sweep value for Vd. For DC only sweeps, VdStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VdStep	(double) The number of steps for the Vd sweep. (Max = 10000)
GateRange	(int) The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
GateCompliance	(double) The current compliance for the DUT Gate.
DrainRange	(int) The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA

12 Limited Auto 100 mA

DrainCompliance	(double) The current compliance for the DUT Drain.
NPLC	(double) The DC measurement integration time in NPLC (Number of Power Line cycles).
SweepDelay	(double) Time, in seconds, between the DC source and measure for each sweep point.
MaxIg	(double) The max current allowed on the DUT Gate. If the Gate current becomes greater than MaxIg, the test will exit.
MaxId	(double) The max current allowed on the DUT Drain. If the Drain current becomes greater than MaxId, the test will exit.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the gate power becomes greater than the MaxPowerGate, the test will exit.
MaxPowerDrain	(double) The max power allowed on the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the test will exit.
GateSMU	(char*) String representing the ID for the SMU connected to the DUT Gate terminal.
DrainSMU	(char*) String representing the ID for the SMU connected to the DUT Drain terminal.
GateVPU	(char*) String representing the ID for the PG2 connected to the DUT Gate terminal.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.
DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
DrainVSize DrainISize GateVSize GateISize	(int) Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.
PostData	(int) Turns on real time graphing. (1 = Real Time Graphing, 0 = No Real Time Graphing)

Outputs:

DrainV	(double) Array of programmed drain voltage values.
DrainI	(double) Array of measured drain currents.
GateV	(double) Array of measured gate voltages.
GateI	(double) Array of measured gate currents.

Vd_Id_Single_Pulse_QPulseIV

Module Return Type: int
Number of Parameters: 37

Arguments:

Vgs,double,Input,2,,
VgQPoint,double,Input,0,,
VdStart,double,Input,0,,
VdStop,double,Input,4,,
VdStep,double,Input,.05,,
VdQPoint,double,Input,0,,
PulseWidth,double,Input,300e-9,,
PulsePeriod,double,Input,1e-6,,
RiseTime,double,Input,100e-9,,
FallTime,double,Input,100e-9,,
Average,int,Input,100,,
GateVPURange,double,Input,5,,
DrainVPURange,double,Input,5,,
GateScpRange,double,Input,0,,
DrainScpRange,double,Input,0,,
GateCompliance,double,Input,.1,,
DrainCompliance,double,Input,.1,,
GateLoadLine,int,Input,1,,
DrainLoadLine,int,Input,1,,
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV,D_ARRAY_T,Output,,,
DrainVSize,int,Input,10000,,
DrainI,D_ARRAY_T,Output,,,
DrainISize,int,Input,10000,,
GateV,D_ARRAY_T,Output,,,
GateVSize,int,Input,10000,,
GateI,D_ARRAY_T,Output,,,
GateISize,int,Input,10000,,
PostData,int,Input,1,,

Description:

The Vd_Id_Single_Pulse_QPulseIV sweep is used to perform a single pulsed Vd-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical DC Vd-Id but only two sources are

used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2 channel scope, Model 4200-SCP2HR.

To create a family of curves, either change Vgs and run the test using append or use the Vd_Id_FET_Family_QPulseIV function.

Inputs:

Vgs	(double) The pulsed Gate-Source voltage bias. Vgs is range dependent and can be between -5 V to +5 V or between -20 V to +20 V.
VgQPoint	(double) The base, or bias point, of the pulsed Gate source. VgQPoint is range dependent and can be between -5 V to +5 V or between -20 V to +20 V.
VdStart	(double) The starting sweep value for Vd. VdStart is range dependent and must be between -10 V to +10 V or between -40 V to +40 V.
VdStop	(double) The final sweep value for Vd. VdStop is range dependent and must be between-10 V to +10 V or between -40 V to +40 V.
VdStep	(double) The number of steps for the Vd sweep. (Max = 10000)
VdQPoint	(double) The base value, or bias point, of the pulsed drain sweep. VdQPoint is range dependent and must be between -10 V to +10 V or -40 V to +40 V.
PulseWidth	(double) The Vgs and Vds pulse width (PW). The Pulse Width range: 300 ns to (1s - 10 ns) in 10 ns resolution steps.
PulsePeriod	(double) The pulse period for Vgs and Vds. Minimum period is (FallTime/2) + (RiseTime/2) + PulseWidth + 10 ns. Maximum period is 1s.
RiseTime	(double) The transition time from the Qpoint to the pulse value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
FallTime	(double) The transition time from the pulse to the Qpoint value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
Average	(int) The number of pulses to average (recommend 100 for a reasonable result, use larger for lower current measurements).
GateVPURange	(double) The source range for gate side PG2. Valid ranges are 5 V (High Speed) and 20 V (High Voltage).
DrainVPURange	(double) The source range for the drain side Model 4205-PCU. Valid ranges are 10 V (High Speed) and 40 V (High Voltage).
GateScpRange	(double) The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
DrainScpRange	(double) The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).

GateCompliance	(double) The current compliance for the pulse source on the DUT Gate. This supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load. The compliance can be used to protect the DUT.
DrainCompliance	(double) The current compliance for the pulse source on the DUT Drain. This supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load. The compliance can be used to protect the DUT.
GateLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Gate. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
DrainLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
MaxIg	(double) The max current allowed for the DUT Gate. If the Gate current becomes greater than MaxIg, the test will exit.
MaxId	(double) The max current allowed for the DUT Drain. If the Drain current becomes greater than MaxId, the test will exit.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the Gate power becomes greater than the MaxPowerGate, the test will exit.
MaxPowerDrain	(double) The max power allowed on the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the test will exit.
GateSMU	(char*) String representing the ID for the SMU connected to the DUT Gate terminal.
DrainSMU	(char*) String representing the ID for the SMU connected to the DUT Drain terminal.
GateVPU	(char*) String representing the ID for the PG2 connected to the DUT Gate terminal.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.
DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
DrainVSize DrainISize GateVSize GateISize	(int) Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.
PostData	(int) Turns on real time graphing. (1 = Real Time Graphing, 0 = No Real Time Graphing)

Outputs:

DrainV	(double) Array of programmed drain voltage values.
DrainI	(double) Array of measured drain currents.
GateV	(double) Array of measured gate voltages.
GateI	(double) Array of measured gate currents.

Vg_Id_Pulse_DC_QPulseIV

Module Return Type: int
Number of Parameters: 49

Arguments:

Vd,double,Input,2,,
VgStart,double,Input,0,,
VgStop,double,Input,4,,
VgStep,double,Input,.05,,
VgQPoint,double,Input,0,,
VdQPoint,double,Input,0,,
PulseWidth,double,Input,300e-9,,
PulsePeriod,double,Input,1e-6,,
RiseTime,double,Input,100e-9,,
FallTime,double,Input,100e-9,,
AverageNum,int,Input,100,,
GateVPURange,double,Input,5,,
DrainVPURange,double,Input,5,,
GateSMURange,int,Input,1,,
DrainSMURange,int,Input,1,,
GateScpRange,double,Input,0,,
DrainScpRange,double,Input,0,,
GateLoadLineCorrection,int,Input,1,,
DrainLoadLineCorrection,int,Input,1,,
GateCompliance,double,Input,.1,,
DrainCompliance,double,Input,.1,,
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
NPLC,double,Input,.01,,
DCSourceDelay,double,Input,0,,
DC_vs_Pulse,int,Input,2,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV_DC,D_ARRAY_T,Output,,,

DrainV_DC_Size,int,Input,10000,,
DrainI_DC,D_ARRAY_T,Output,,,
DrainI_DC_Size,int,Input,10000,,
GateV_DC,D_ARRAY_T,Output,,,
GateV_DC_Size,int,Input,10000,,
GateI_DC,D_ARRAY_T,Output,,,
GateI_DC_Size,int,Input,10000,,
DrainV_Pulse,D_ARRAY_T,Output,,,
DrainV_Pulse_Size,int,Input,10000,,
DrainI_Pulse,D_ARRAY_T,Output,,,
DrainI_Pulse_Size,int,Input,10000,,
GateV_Pulse,D_ARRAY_T,Output,,,
GateV_Pulse_Size,int,Input,10000,,
GateI_Pulse,D_ARRAY_T,Output,,,
GateI_Pulse_Size,int,Input,10000,,

Description:

The Vg_Id_Pulse_DC_QPulseIV sweep is used to perform a Pulsed vs. DC Vg-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical Vg-Id but only two sources are used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2 channel scope, Model 4200-SCP2HR.

This routine can run the sweeps in three different ways: 1) DC only; 2) Pulse only; 3) Pulse and DC curves.

All voltage levels specified below assume a 50 Ω DUT load.

Inputs:

Vds	(double) The voltage value for Vd. For DC only sweeps, Vds must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, Vds must be between -10 V to +10 V or -40 V to +40 V depending on the specified source range for the Model 4205-PCU.
VgStart	(double) The starting sweep value for Vg. For DC only sweeps, VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V or -20 V to +20 V depending on the specified source range for the PG2.
VgStop	(double) The final sweep value for Vg. For DC only sweeps, VgStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V or -20 V to +20 V depending on the specified source range for the PG2.
VgNumSteps	(double) The number of steps in the Vg sweep (Max = 10 000).
VgQPoint	(double) The base value, or bias point, of the pulsed gate sweep. VgQPoint is range dependent and must be between -5 V to +5 V or -20 V to +20 V.
VdQPoint	(double) The base value, or bias point, of the pulsed drain source. VdQPoint is range dependent and must be between -10 V to +10 V or -40 V to +40 V.
PulseWidth	(double) The Vgs and Vds pulse width (PW). The Pulse Width range: 300 ns to (1 s - 10 ns) in 10 ns resolution steps.

PulsePeriod	(double) The pulse period for Vgs and Vds. Minimum period is (FallTime/2) + (RiseTime/2) + PulseWidth + 10 ns. Maximum period is 1 s.
RiseTime	(double) The transition time from the Qpoint to the pulse value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
FallTime	(double) The transition time from the pulse to the Qpoint value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
Average	(int) The number of pulses to average (recommend 100 for a reasonable result, use larger for lower current measurements).
GateVPURange	(double) The source range for gate side PG2. Valid ranges are 5 V (High Speed) and 20 V (High Voltage).
DrainVPURange	(double) The source range for the drain side Model 4205-PCU. Valid ranges are 10 V (High Speed) and 40 V (High Voltage).
GateSMURange	(int) The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents: 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
DrainSMURange	(int) The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents: 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA

	12 Limited Auto 100 mA
GateScpRange	(double) The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
DrainScpRange	(double) The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
GateLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Gate. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
DrainLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
GateCompliance	(double) The current compliance for the DUT Gate. For pulse current compliance, the supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
DrainCompliance	(double) The current compliance for DUT Drain. For pulse current compliance, the supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load.
MaxIg	(double) The max current allowed for the DUT Gate, for both DC and pulse sweeps. If the Gate current becomes greater than MaxIg, the present sweep will stop and the test will continue with the next Vg step.
MaxId	(double) The max current allowed on the DUT Drain, for both DC and pulse sweeps. If the Drain current becomes greater than MaxId, the present sweep will stop and the test will continue with the next Vg step.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the Gate power becomes greater than the MaxPowerGate, the present sweep will stop and the test will continue with the next Vg step.
MaxPowerDrain	(double) The max power allowed to the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the present sweep will stop and the test will continue with the next Vg step.
NPLC	(double) The DC measurement integration time in NPLC (Number of Power Line cycles).
DCSweepDelay	(double) Time, in seconds, between the DC source and measure for each sweep point.
DC_vs_Pulse	(int) Determines whether to run a DC and Pulse test or a DC only test or a Pulse only test.0 - Pulse Only, 1 - DC Only, 2 - DC and Pulse.
GateSMU	(char*) String representing the ID for the SMU connected to the gate.
DrainSMU	(char*) String representing the ID for the SMU connected to the drain.
GateVPU	(char*) String representing the ID for the PG2 connected to the gate.

DrainVPUHigh (char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.

DrainVPULow (char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.

DrainV_DC_Size
DrainI_DC_Size
GateV_DC_X_Size
GateI_DC_X_Size
DrainV_Pulse_Size
DrainI_Pulse_Size
GateV_Pulse_X_Size
GateI_Pulse_X_Size
(int) Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.

Outputs:

DrainV_DC
DrainV_Pulse
(double) Array of measured drain voltage values.

DrainI_DC
DrainI_Pulse
(double) Array of measured drain currents.

GateV_DC_X
GateV_Pulse_X
(double) Array of programmed gate voltages.

GateI_DC_X
GateI_Pulse_X
(double) Array of measured gate currents.

Vg_Id_Single_DC_QPulseIV

Module Return Type: int
Number of Parameters: 28

Arguments:

Vds,double,Input,3,,
VgStart,double,Input,0,,
VgStop,double,Input,2,,
VgStep,double,Input,.05,,
GateRange,int,Input,1,,
GateCompliance,double,Input,.1,,
DrainRange,int,Input,1,,
DrainCompliance,double,Input,.1,,
NPLC,double,Input,.01,,
SweepDelay,double,Input,0,,
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,

```
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV,D_ARRAY_T,Output,,,
DrainVSize,int,Input,10000,,
DrainI,D_ARRAY_T,Output,,,
DrainISize,int,Input,10000,,
GateV,D_ARRAY_T,Output,,,
GateVSize,int,Input,10000,,
GateI,D_ARRAY_T,Output,,,
GateISize,int,Input,10000,,
PostData,int,Input,1,,
```

Description:

The Vg_Id_Single_DC_QPulseIV sweep is used to perform a DC Vg-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical Vg-Id but only two sources are used: one for the DUT Gate and one for the DUT Drain.

All voltage levels specified below assume a 50 Ω DUT load.

Inputs:

Vds	(double) The voltage value for Vd. Vds must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VgStart	(double) The starting sweep value for Vg. VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VgStop	(double) The final sweep value for Vg. VgStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VgStep	(double) The number of steps in the Vg sweep (Max = 10 000).
GateRange	(int) The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents: 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
GateCompliance	(double) The current compliance for the DUT Gate.
DrainRange	(int) The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents: 1 Full Auto

2 Limited Auto 10 pA
3 Limited Auto 100 pA
4 Limited Auto 1 nA
5 Limited Auto 10 nA
6 Limited Auto 100 nA
7 Limited Auto 1 uA
8 Limited Auto 10 uA
9 Limited Auto 100 uA
10 Limited Auto 1 mA
11 Limited Auto 10 mA
12 Limited Auto 100 mA

DrainCompliance	(double) The current compliance for the DUT Drain.
NPLC	(double) The DC measurement integration time in NPLC (Number of Power Line cycles).
DCSweepDelay	(double) Time, in seconds, between the DC source and measure for each sweep point.
MaxIg	(double) The max current allowed on the DUT Gate. If the Gate current becomes greater than MaxIg, the test will exit.
MaxId	(double) The max current allowed on the DUT Drain. If the Drain current becomes greater than MaxId, the test will exit.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the gate power becomes greater than the MaxPowerGate, the test will exit.
MaxPowerDrain	(double) The max power allowed on the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the test will exit.
GateSMU	(char*) String representing the ID for the SMU connected to the gate.
DrainSMU	(char*) String representing the ID for the SMU connected to the drain.
GateVPU	(char*) String representing the ID for the PG2 connected to the gate.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.
DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
DrainVSize GateISize	(int) Sizes of the output arrays. Note that DrainISize all arrays should be the GateISize same size and need GateVSize to be large enough to hold all sweep points.
PostData	(int) Turns on real time graphing. (1 = Real Time Graphing, 0 = No Real Time Graphing)

Outputs:

DrainV	(double) Array of measured drain voltage values.
DrainI	(double) Array of measured drain currents.
GateV	(double) Array of programmed gate voltages.
GateI	(double) Array of measured gate currents.

Vg_Id_Single_Pulse_QPulseIV

Module Return Type: int
Number of Parameters: 37

Arguments:

Vds,double,Input,3,,
VdQPoint,double,Input,0,,
VgStart,double,Input,0,,
VgStop,double,Input,2,,
VgStep,double,Input,.05,,
VgQPoint,double,Input,0,,
PulseWidth,double,Input,300e-9,,
PulsePeriod,double,Input,1e-6,,
RiseTime,double,Input,100e-9,,
FallTime,double,Input,100e-9,,
Average,int,Input,10,,
GateVPURange,double,Input,5,,
DrainVPURange,double,Input,5,,
GateScpRange,double,Input,0,,
DrainScpRange,double,Input,0,,
GateCompliance,double,Input,.1,,
DrainCompliance,double,Input,.1,,
GateLoadLine,int,Input,1,,
DrainLoadLine,int,Input,1,,
MaxIg,double,Input,1,,
MaxId,double,Input,1,,
MaxPowerGate,double,Input,200,,
MaxPowerDrain,double,Input,200,,
GateSMU,char *,Input,"SMU1",,
DrainSMU,char *,Input,"SMU2",,
GateVPU,char *,Input,"VPU1",,
DrainVPUHigh,char *,Input,"VPU2",,
DrainVPULow,char *,Input,"VPU3",,
DrainV,D_ARRAY_T,Output,,,
DrainVSize,int,Input,10000,,
DrainI,D_ARRAY_T,Output,,,
DrainISize,int,Input,10000,,
GateV,D_ARRAY_T,Output,,,
GateVSize,int,Input,10000,,
GateI,D_ARRAY_T,Output,,,
GateISize,int,Input,10000,,
PostData,int,Input,1,,

Description:

The Vg_Id_Single_Pulse_QPulseIV sweep is used to perform a Pulsed Vg-Id sweep using the Model 4200-PIV-Q package. This test is similar to a typical Vg-Id but only two sources are used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2 channel scope, Model 4200-SCP2HR.

All voltage levels specified below assume a 50 Ω DUT load.

Inputs:

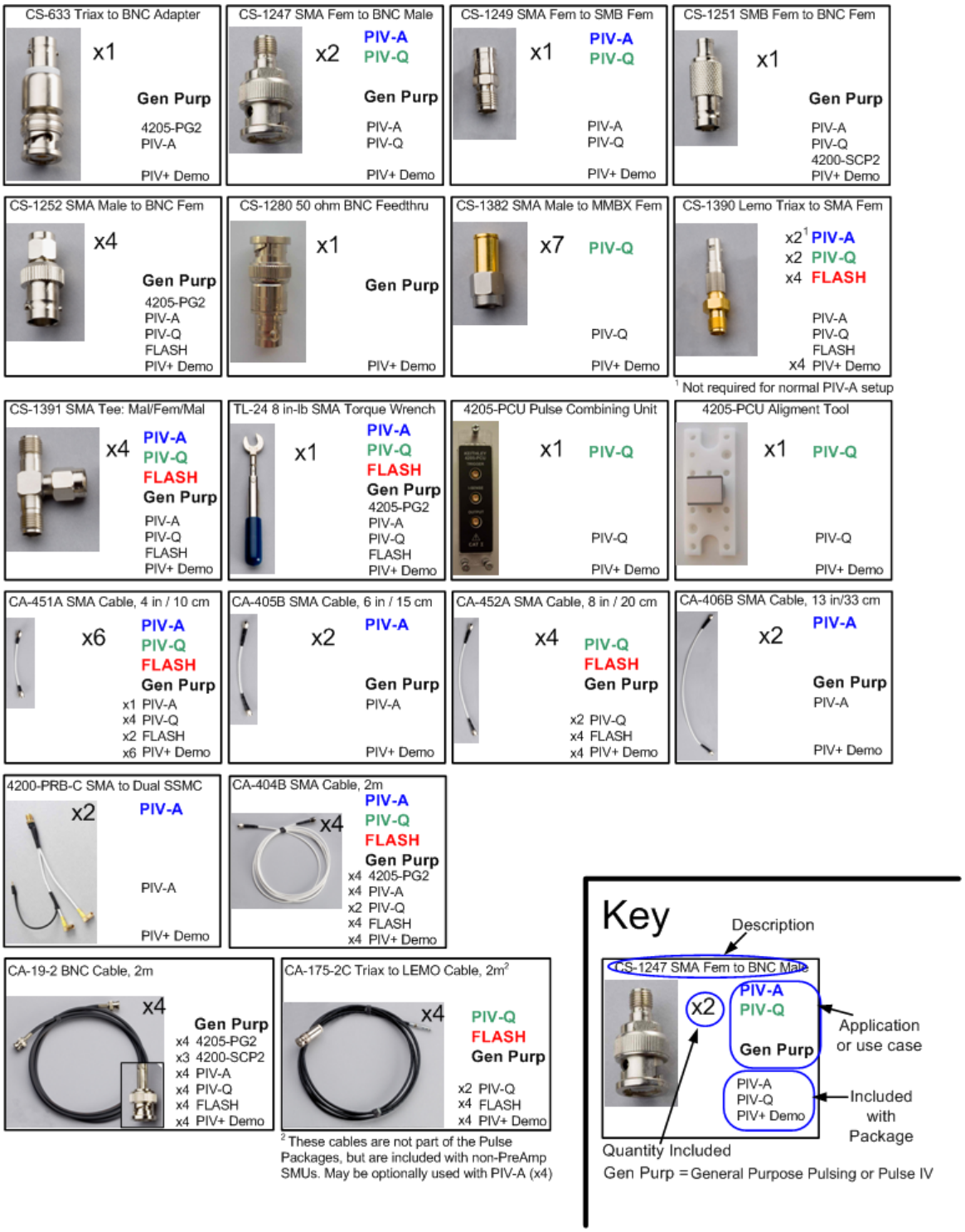
Vds	(double) The voltage value for Vd. Vds must be between -10 V to +10 V or -40 V to +40 V depending on the specified source range for the Model 4205-PCU.
VdQPoint	(double) The base value, or bias point, of the pulsed drain source. VdQPoint is range dependent and must be between -10 V to +10 V or -40 V to +40 V.
VgStart	(double) The starting sweep value for Vg. VgStart is range dependent and must be between -5 V to +5 V or between -20 V to +20 V.
VgStop	(double) The final sweep value for Vg. VgStop is range dependent and must be between -5 V to +5 V or between -20 V to +20 V.
VgStep	(double) The number of steps in the Vg sweep (Max = 10 000).
VgQPoint	(double) The base value, or bias point, of the pulsed gate sweep. VgQPoint is range dependent and must be between -5 V to +5 V or -20 V to +20 V.
PulseWidth	(double) The Vgs and Vds pulse width (PW). The Pulse Width can be 300 ns to 1s (10 ns resolution).
PulsePeriod	(double) The pulse period for Vgs and Vds. Minimum period is (FallTime/2) + (RiseTime/2) + PulseWidth + 10 ns. Maximum period is 1s.
RiseTime	(double) The transition time from the Qpoint to the current pulse value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (although recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and fall time.
FallTime	(double) The transition time from the current value to the QPoint base value for both Vgs and Vds. The transition time is source range dependent. For the 5 V range (10 V for Model 4205-PCU) the minimum transition time is 10 ns (although recommended minimum is 50 ns) and for the 20 V range (40 V for the Model 4205-PCU) the minimum transition time is 100 ns. The maximum transition time is dependent on the pulse width, period, and rise time.
Average	(int) The number of pulses to average (recommend 100 for a reasonable result, use larger for lower current measurements).
GateVPURange	(double) The source range for gate side PG2. Valid ranges are 5 V (High Speed) and 20 V (High Voltage).
DrainVPURange	(double) The source range for the drain side Model 4205-PCU. Valid ranges are 10 V (High Speed) and 40 V (High Voltage).
GateScpRange	(double) The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
DrainScpRange	(double) The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
GateCompliance	(double) The current compliance for the pulse source on the DUT Gate. This supplied value is used to calculate the maximum voltage to source to the DUT, based on a default 50 Ω DUT load. The compliance can be used to protect the DUT.
DrainCompliance	(double) The current compliance for the pulse source on the DUT Drain. This supplied value is used to calculate the maximum voltage to source to the

	DUT, based on a default 50 Ω DUT load. The compliance can be used to protect the DUT.
GateLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Gate. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
DrainLoadLine	(int) Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced. (1 = Use Load Line, 0 = No Load Line).
MaxIg	(double) The max current allowed for the DUT Gate, for both DC and pulse sweeps. If the Gate current becomes greater than MaxIg, the present sweep will stop and the test will continue with the next Vg step.
MaxId	(double) The max current allowed on the DUT Drain, for both DC and pulse sweeps. If the Drain current becomes greater than MaxId, the present sweep will stop and the test will continue with the next Vg step.
MaxPowerGate	(double) The max power allowed to the DUT Gate. If the Gate power becomes greater than the MaxPowerGate, the test will exit.
MaxPowerDrain	(double) The max power allowed to the DUT Drain. If the Drain power becomes greater than the MaxPowerDrain, the test will exit.
GateSMU	(char*) String representing the ID for the SMU connected to the gate.
DrainSMU	(char*) String representing the ID for the SMU connected to the drain.
GateVPU	(char*) String representing the ID for the PG2 connected to the gate.
DrainVPUHigh	(char*) String representing the top VPU on the Model 4205-PCU. This is the card on the right (or the card with the lower ID number), when facing the back of the system.
DrainVPULow	(char*) String representing the bottom VPU on the Model 4205-PCU. This is the card on the left (or the card with the higher ID number), when facing the back of the system.
DrainVSize DrainISize GateVSize GateISize	(int) Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.
PostData	(int) Turns on real time graphing. (1 = Real Time Graphing, 0 = No Real Time Graphing)
Outputs:	
DrainV	(double) Array of measured drain voltage values.
DrainI	(double) Array of measured drain currents.
GateV	(double) Array of programmed gate voltages.
GateI	(double) Array of measured gate currents.

Pulse adapters, cables, hardware and PCU

The various adapters, cables and hardware used for the pulse projects are shown in Figure 11-39. The 4205-PCU that is used for the PIV-Q package is also shown.

Figure 11-39
Pulse adapters, cables and hardware



Section 12

Pulse Projects for Models 4200-PIV-A and 4200-PIV-Q

In this section:

Model 4205-RBT (Remote Bias Tee) and Power Divider

The Model 4205-RBT and Power Divider are used for the Keithley PulseIV-Complete and Demo-PulseIV projects. Two RBT Bias Tee adapters and one 3-Port Power Divider are included with the Models 4200-PIV-A and 4200-PIV-HR Pulse-IV solution bundles. Also included are two Model 4200-MAG-BASE mounts (magnetically attaches Bias Tee adapters to platen of prober).

NOTE *Refer to Section 4 (Pulse Applications) of the Applications Manual for complete connection details using the entire PIV package to test a transistor DUT.*

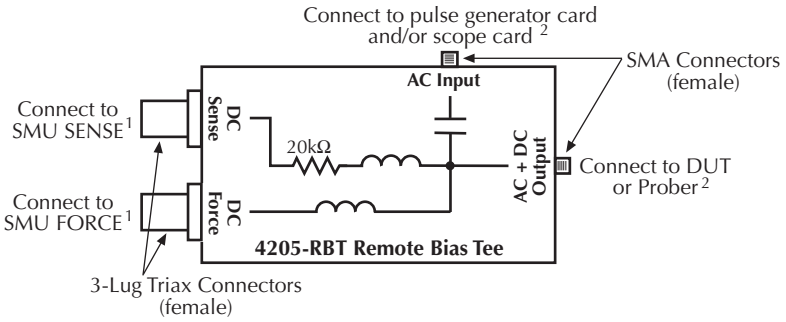
RBT

The Remote Bias Tee (RBT) adapter (refer to Figure 12-1) is a coupler for DC bias from a SMU and pulse output from a Model 4205-PG2 pulse generator channel. The output of the RBT provides pulse output riding on the DCV bias.

As shown in Figure 12-1, the RBT has two three-lug female triax connectors for connection to an SMU (FORCE and SENSE), and two female SMA connectors; one for AC inputs, such as a pulse generator card and scope card (Model 4200-SCP2HR or 4200-SCP2), and one for AC+DC output connection to a prober or directly to a DUT.

Figure 12-1 also shows the simplified schematic of the RBT. The capacitor allows pulses from the pulse generator card to pass through to the output, while blocking DC from the SMU. The inductors allow DC from the SMU to pass through to the output, while blocking pulses from the pulse generator.

Figure 12-1
Model 4205-RBT



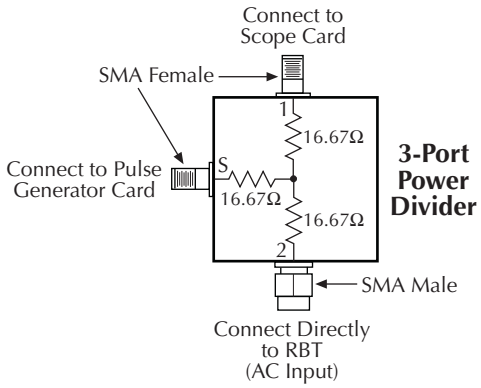
1) When using a SMU PreAmp, use 4200-TRX-X cables for connections. When NOT using a PreAmp, use 4200-MTRX-X cables for connections.
2) Use SMA cables (male-to-male) for connections.

3-Port Power Divider

The 3-Port Power Divider divides the electrical power equally among its three connectors using a 16.67W resistor in each leg (see Figure 12-2). The Power Divider is used on the gate of a FET to provide an impedance matched signal (pulse) path (50 Ω).

As shown in Figure 12-2, the Power Divider is equipped with two SMA female connectors and one SMA male connector. The SMA male connector allows the Power Divider to connect directly to the RBT (AC Input).

Figure 12-2
3-Port Power Divider



Using an RBT and Power Divider

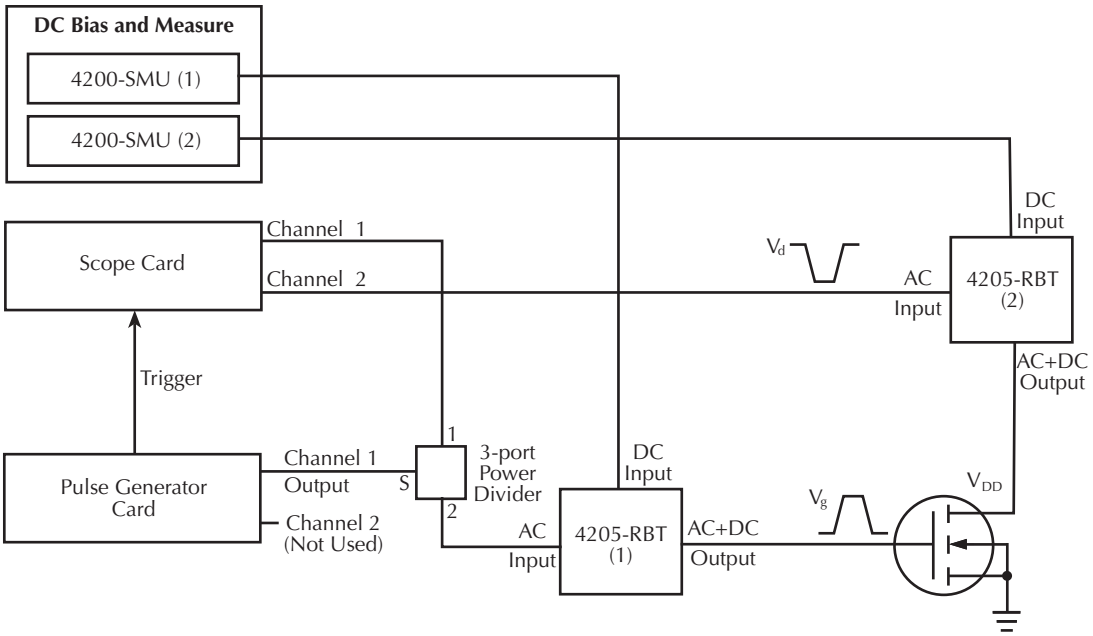
Figure 12-3 shows a block diagram of the PIV (pulse IV) test system that uses two SMUs, a pulse generator card (one channel), a scope card (both channels), two RBTs and the Power Divider.

The Power Divider provides impedance matching and an RBT functions as a coupler for DC bias from a SMU and pulse output (AC) from a pulse generator card. The output of an RBT provides pulse output that rides on the DC bias level. The scope card is used to capture pulse waveforms or pulse readings. The DUT (device under test) is typically a wafer site (via prober) or a discrete device.

The capacitor for a RBT functions as a low-impedance component for high speed pulses, and as a high-impedance element for DC. This allows the high-speed pulses from the pulse generator card to pass through to the output, while blocking DC from the SMU.

The inductors of an RBT function as low-impedance components for DC, and as high-impedance components for high speed pulses. This allows the DC bias from the SMU to pass through to the output, while blocking the high speed pulses from the pulse generator.

Figure 12-3
Block diagram – PIV test system



PulseIV-Complete and Demo-PulseIV projects

The PulseIV-Complete and Demo-PulseIV projects provide PIV (pulse IV) testing. These projects are included with the Model 4200-PIV-A and 4200-PIV-HR packages.

NOTE *See Pulse IV in Section 4 of the Applications Manual for details on connecting and running the PIV package tests.*

Overall 4200 Pulse IV capabilities

- Pulse voltage on gate from -5 to +5 V, zero referenced
- Pulse widths of 40-150 ns (due to RBT), adjustable in 10 ns increments
- Periods of 40 us and larger (due to RBT), adjustable in 10 ns increments. The maximum 3-Port Power Divider duty cycle is 0.1%.
- Pulse transition is programmed to 10 ns, which results in a 13 ns transition time.
- DC voltage bias on drain, provided by SMU, from -210 V to +210 V.
- Drain current pulse measurement, up to 100 mA with 5 uA resolution (better resolution available by averaging multiple pulses) provided by 8 bit scope card.
- Vds-Id and Vgs-Id sweeps
- Single pulse scope shot for setup validation and transient testing

Note that the above list is specific to the operation of the entire PIV package. The individual components, such as the pulse generator card and scope card, may have different capabilities. For example, the Model 4205-PG2 pulse generator card can be programmed to output a 10 ns wide pulse, but this pulse is not sufficient for a Pulse-IV measurement and is therefore not permitted in the supplied PIV setup.

4200 Project: PulseIV-Complete

This project includes all pulse source and pulse measure tests for the 4200-PIV package. This project is used for both testing on customer devices and demonstration of the 4200-PIV package.

The PulseIV-Complete project has a two Initialization steps for pulse calibration and nine main tests under the device 4terminal-n-fet. This project permits comparison between DC and pulse IV sweeps. These nine primary tests consist of three sets. The first test in each set is the DC IV sweep, with the second test using pulse IV for the sweep. The third test provides both DC and Pulse IV sweeps combined into a single UTM. The first set is Vds-id, the second is Vgs-id and the third is another Vds-id test, but with voltages for both the gate and drain increased to demonstrate self-heating on a device. The last test, scope-shot, provides a snapshot of the pulse waveforms that are used during the Pulse IV tests. Please note that the values seen in the scopeshot test are approximate values, although calibrated measurements are also available.

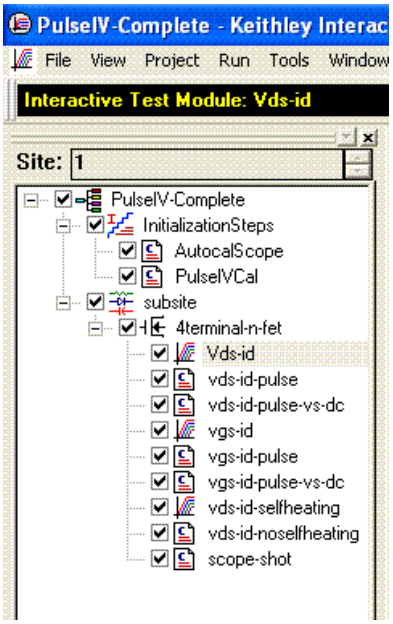
This project supports both nMOS and pMOS testing, just use the appropriate voltage sign for sourcing the proper voltage polarity.

The tests for the PulseIV-Complete project are shown in the Project View in Figure 12-4. The initialization test are described below. See "PIV tests" on page 12-9 for descriptions of the Pulse IV tests.

Initialization Tests

- AutocalScope – Runs the self calibration and offset measurement routine for the scope card. This routine should be run before every PulseIV cal and periodically to capture any drift in the scope card. It requires all connections be removed from the scope card and takes less than one minute.
- PulseIVCal - This is the Pulse-IV cable compensation routine that should be used during initial setup and whenever interconnects are changed. This routine takes about 3-4 minutes and requires changing the connection at the mid-point to transition from an open to a through (or short) connection. This routine is similar to an open/short cal used for capacitance measurements.

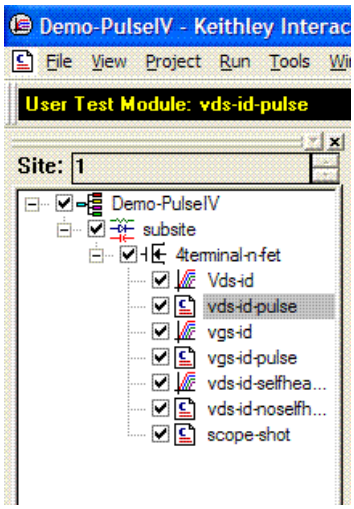
Figure 12-4
PulseIV-Complete project plan



Demo-PulseIV tests

The Demo-PulseIV project (shown in Figure 12-5) uses same PIV tests that are used by the PulseIV-Complete project. However, this project does not contain initialization tests. Also, the demo tests are simplified versions of the complete tests in which some parameters were removed and not allowed to be set by the user.

Figure 12-5
Demo-PulseIV project plan



Test configuration and instrumentation

The test configuration for the PIV projects is shown by the block diagram in Figure 12-3. This test configuration will accommodate all the tests used in the PIV projects. The primary components of the test system are summarized as follows:

Pulse generator card – The pulse generator card is an internal instrument and has two pulse output channels. See the User's Manual, Pulse cards, page 1-25 for details.

Scope card – The scope card is an internal instrument that has two input channels. See the User's Manual, SCP2 (Oscilloscope), page 1-32 for details.

RBTs – The RBT (Remote Bias Tee) function as a coupler for AC (pulse) and DC signals. See Model 4205-RBT (Remote Bias Tee) and Power Divider for details.

See Pulse IV in Section 4 of the Model 4200 Applications Manual for details on connections and running the tests.

NOTE *For PIV project configurations, make sure to disable Model 4200-SCS high voltage. This will prevent SMU voltages greater than 42V from being applied to the device under test or in a fixture. For details, see Safety interlock connections in Section 4.*

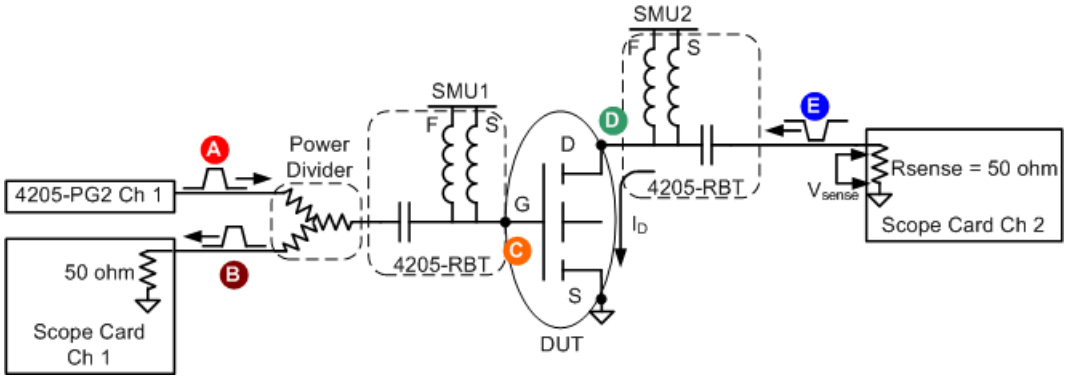
Theory of operation for the 4200-PIV-A package

In general, the PIV package applies a pulse to the gate of the DUT (see Figure 12-6), while DC biasing the drain. The source and body connections are connected to ground/shield. The dual-channel scope card measures the gate voltage and drain voltage. The drain current is calculated by the voltage drop across the 50 Ω termination of the scope (Rsense on the scope card channel 2).

NOTE *The configuration and underlying method of operation of the 4200-PIV-A package are roughly based on concepts and work by K.A. Jenkins[1] and A. Kerber[2], with results by C.D. Young[3] and others.*

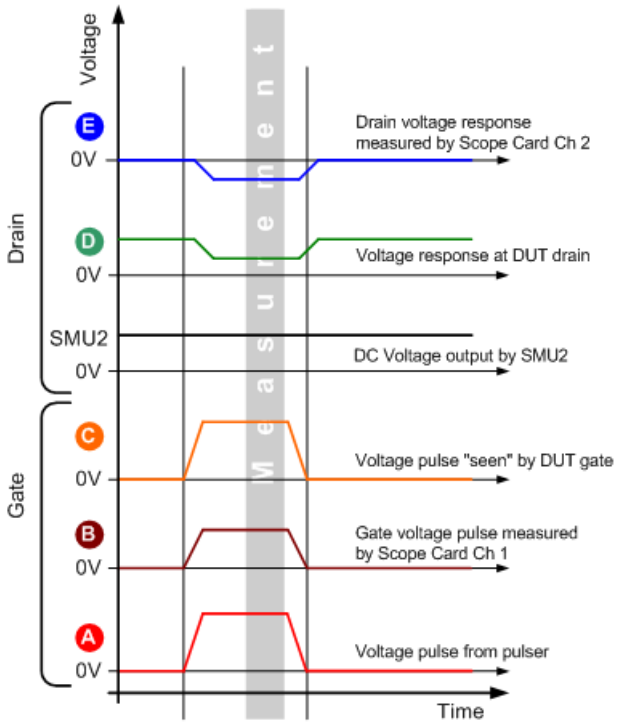
Figure 12-7 shows the corresponding waveforms for the test system. They are from select points to provide snapshots of the signals and to show the effects of various aspects of the setup. Although many waveforms are displayed in Figure 12-7, there are only two waveforms that are actually measured by the scope card (waveforms B and E). The waveforms shown in Figure 1b are a snapshot of the signals shown in Figure 1a, all graphed on the same time scale. Showing all waveforms together permits comparison of the DC offset and relative amplitudes in various parts of the schematic.

Figure 12-6
Pulse IV schematic diagram



1. Jenkins, K.A. Sun, J.Y.-C. Gautier, J., “Characteristics of SOI FET's under pulsed conditions,” *IEEE Transactions on Electron Devices*, vol 44, issue 11, pp. 1923-1930, Nov 1997.
2. A. Kerber, E. Cartier, L. Pantisano, M. Rosmeulen, R. Degraeve, T. Kauerauf, G. Groeseneken, H. E. Maes, and U. Schwalke, "Characterization of the VT instability in SiO2/HfO2 gate dielectrics," Proc. IEEE., IRPS pp. 41, 2003.
3. C. D. Young, R. Choi, J. H. Sim, B. H. Lee, P. Zeitzoff, Y. Zhao, K. Matthews, G. A. Brown, and G. Bersuker, “Interfacial Layer Dependence of HfSixOy Gate Stacks on Vt Instability and Charge Trapping Using Ultra-short Pulse I-V Characterization,” Proc. IEEE IRPS, p. 75-79, 2005.

Figure 12-7
Waveforms at select points in Figure 12-6



At the start of the test, the drain of the device under test (DUT) is biased with SMU2 and is waiting for a gate pulse to turn on the transistor and cause drain current, Id, to flow.

A. Gate Voltage Pulse: This waveform is the output from Channel 1 of the PG2 pulse generator card. Its amplitude is about the same as that seen by the DUT gate (C), except for some small cable losses. Note that there is no DC offset on this waveform, as any DC voltage would cause excessive power to be dissipated in the power divider.

B. Gate voltage pulse measured by the oscilloscope (scope card channel 1): This is the portion of the pulse that is transmitted through the power divider. Because of the divider and the 50 Ω input impedance of the scope, the amplitude of the pulse is reduced 33%, plus any additional cable losses. The PulseIV calibration accounts for this loss caused by the power divider and cabling. The power splitter splits the power in half (3dB), which means that the voltage is lower by the square root of two. A 2V pulse will in an ~1.41V signal (assuming no other losses). Note that this measurement is used to determine the proper magnitude of the V_G pulse, but it does not measure the gate current.

   SMU1 (not shown in Figure 12-7): This is the voltage output by SMU1 during Pulse IV testing. During Pulse IV testing in the PulseIV-Complete project, SMU1 is fixed at 0 V.

C. Voltage pulse at DUT gate: This waveform shows what the DUT gate sees. This is the pulse that turns on the transistor and causes the drain current to flow. The Pulse IV calibration takes into account the losses in the cabling, with the assumption that the gate is high impedance.

   SMU2: DC Voltage bias applied to the drain, through the RBT.

D. Drain response: This waveform shows the DC bias and the resulting response to the gate pulse. Note that the drain is DC biased and the pulse in this waveform is the result of the gate pulse. There is no pulse applied to the drain, rather the pulse shown is the response of the drain current flowing.

E. Drain voltage response measured by oscilloscope (4200-SCP2 Channel 2): The response of the drain is due to the gate pulse turning on the transistor and allowing drain current to flow. This drain current (I_D) flows through the 50 Ω input impedance of the scope, which acts as a current sense resistor (R_{SENSE}), resulting in the voltage drop in the waveform (V_{SENSE} in Figure 12-6). This voltage response is negative due to the direction of current flow through the 50 Ω resistor and the fact that the scope is ground-referenced. Note that this waveform is AC shifted to 0 V, due to the coupling capacitor in the RBT.

PIV tests

Vds-id
vds-id-pulse
vds-id-pulse-vs-dc

The DC **Vds-id** test, shown in Figure 12-8 and Figure 12-9 is a typical family of curves. The drain voltage, Vd, is swept from 0 to 4V and after each Vd sweep the gate voltage, Vg, is stepped to the next value. The key difference between a traditional DC-only Vds-id and the DC IV through the RBTs (remote bias tees) is the number of SMUs. For DC IV through the bias tees, two SMUs are used, with the Source and Body/Bulk connections connected to ground (SMA coax shield), as shown in the schematic in Figure 12-6.

The **Vds-id-pulse** test is shown in Figure 12-10 and Figure 12-11. Since the pulse tests are UTMs (User Test Modules), the parameters are changed via the table interface shown in Figure 12-10. For the pulse Vds-id, the gate voltage is not stepped, so each unique gate voltage must be entered before appending the next Vds-id curve to the graph. In addition to the tabular parameter format, there is a simplified interface for changing the most common parameters. This interface is accessed via the GUI button, as shown in Figure 12-10. Figure 12-11 shows the GUI.

NOTE *Both the DC and pulse test parameters are easily modified to permit interactive investigation of transistor behavior.*

Figure 12-8
DC Vds-id ITM Graph tab

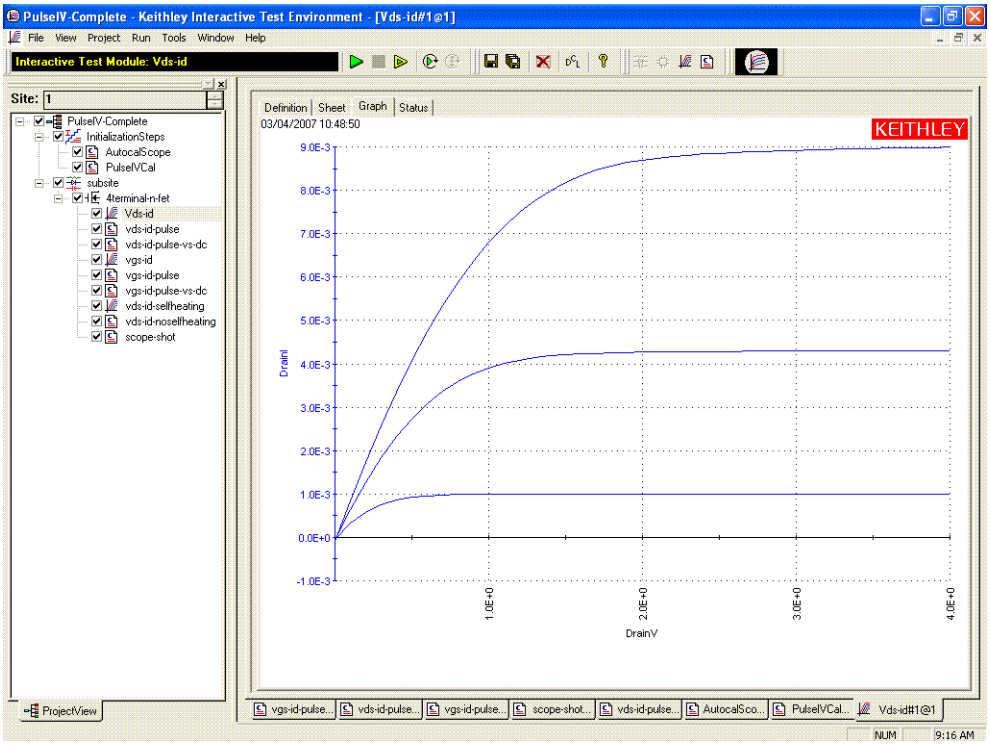


Figure 12-9
Vds-id ITM Definition tab

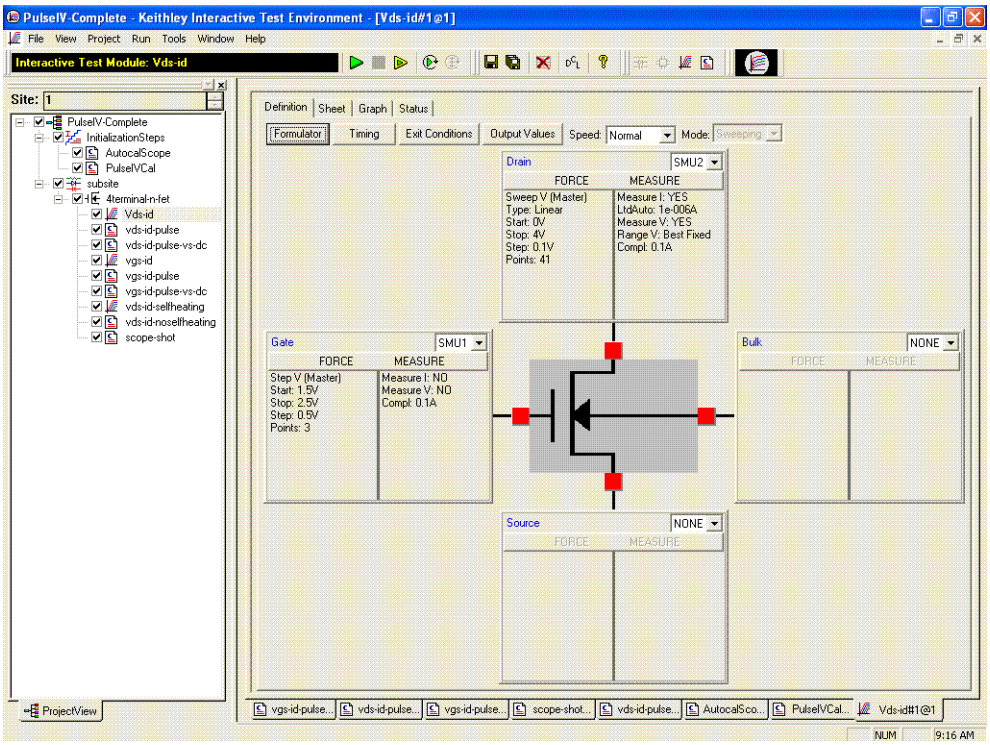


Figure 12-10
vds-id-pulse UTM Definition tab

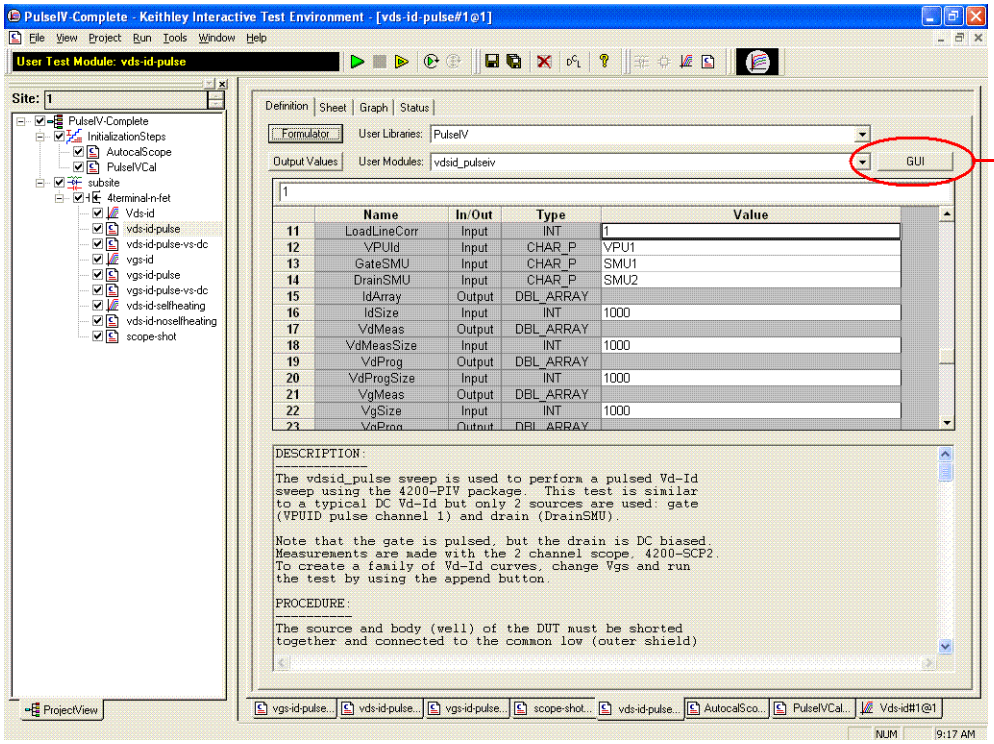
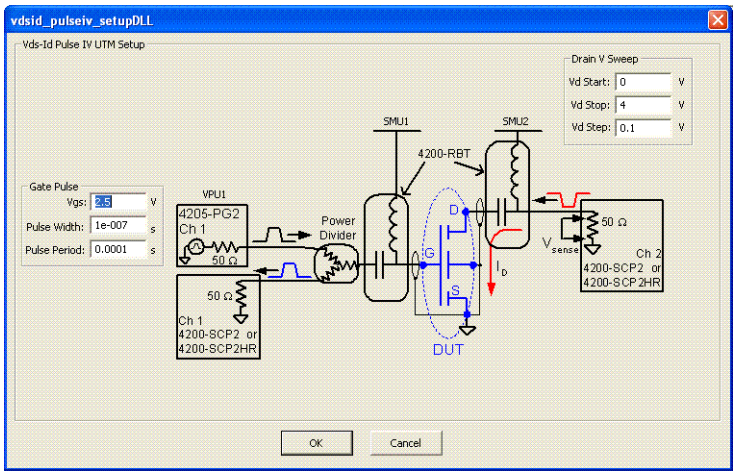


Figure 12-11
Pulse Vds-Id GUI UTM



The GUI for the Vds-Id displays a simplified schematic, as well as the most common parameters. Changing values in this GUI and clicking OK updates the parameter values in the table (see Figure 12-10). The use of GUI is entirely optional. Running the test still requires clicking on the green Run triangle icon at near the top of the window. Note that the values from the definition table are copied into the GUI, so there is never a mis-match between the tabular and GUI parameter values.

A third Vds-id test (**vds-id-pulse-vs-dc**) incorporates both pulse and DC into a single UTM. This is useful for comparing DC and pulse results without having to copy data between tests. Figure 12-12 shows the GUI and Figure 12-13 shows the graph. This combined test shares the same Vg and Vd values, in addition to all of the DC and pulse measurement parameters, which are not on the GUI, but are available on the Definition tab. The test alternates between the DC IV and pulse IV tests,

so each Vg is run first as DC then as pulse before stepping to the next Vg voltage. Note that during the test, neither the graph tab or sheet tab is updated.

Figure 12-12
Pulse and DC Vds-id UTM GUI

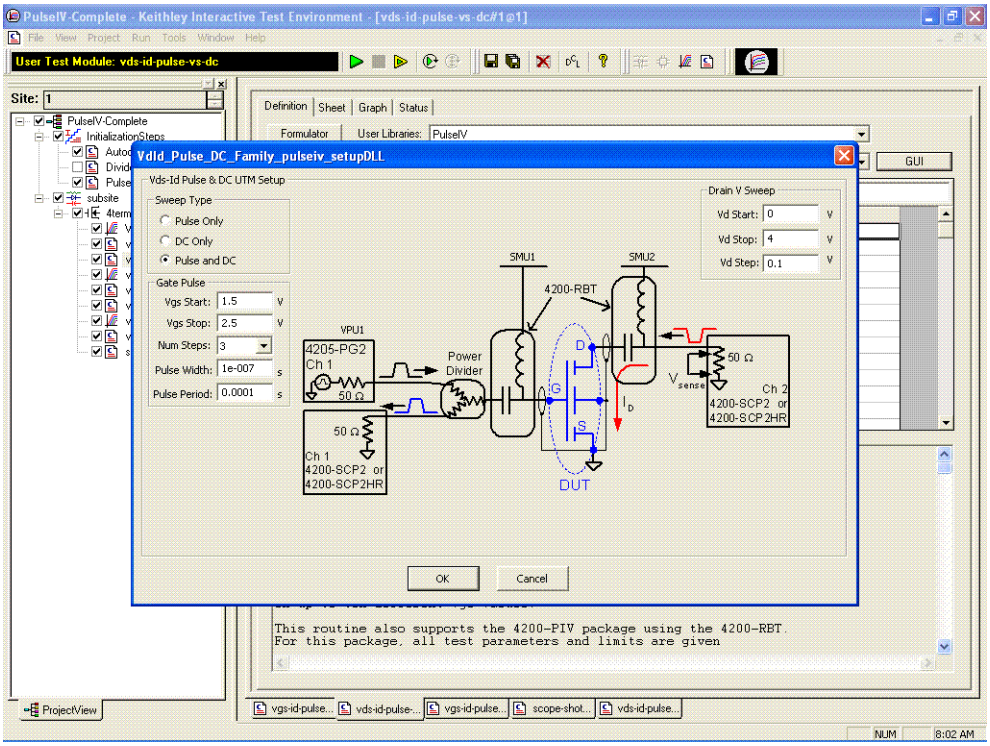
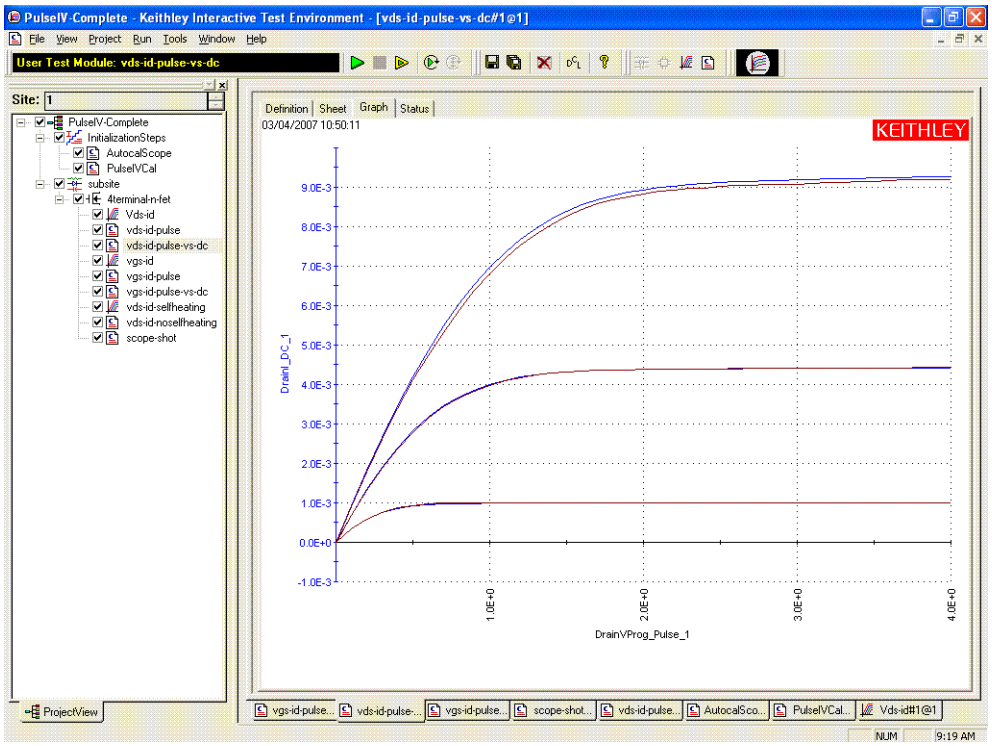


Figure 12-13
Pulse and DC Vds-id UTM Graph tab



vgs-id
vgs-id-pulse
vgs-id-pulse-vs-dc

The DC **vgs-id** test, shown in Figure 12-14 and Figure 12-15. The key difference between a traditional DC-only Vgsid and the DC IV through the RBT (remote bias tees) is the number of SMUs. For DC IV through the bias tees, two SMUs are used, with the Source and Body/Bulk connections connected to ground (SMA coax shield), as shown in the schematic in Figure 12-17.

The **vgs-id-pulse** test is shown in Figure 12-16, Figure 12-17 and Figure 12-18. Since the pulse tests are UTMs (User Test Modules), the parameters are changed via the table interface shown in Figure 12-16 or the GUI shown in Figure 12-17.

Figure 12-14
DC vgs-id ITM Definition tab

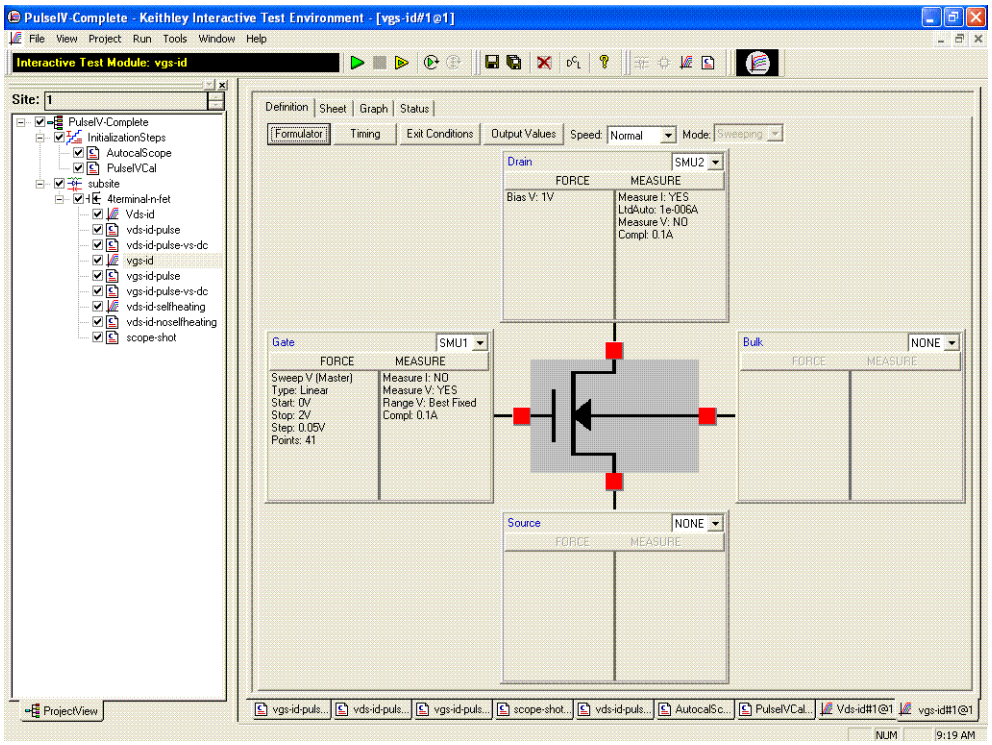


Figure 12-15
DC Vgs-id ITM Graph tab

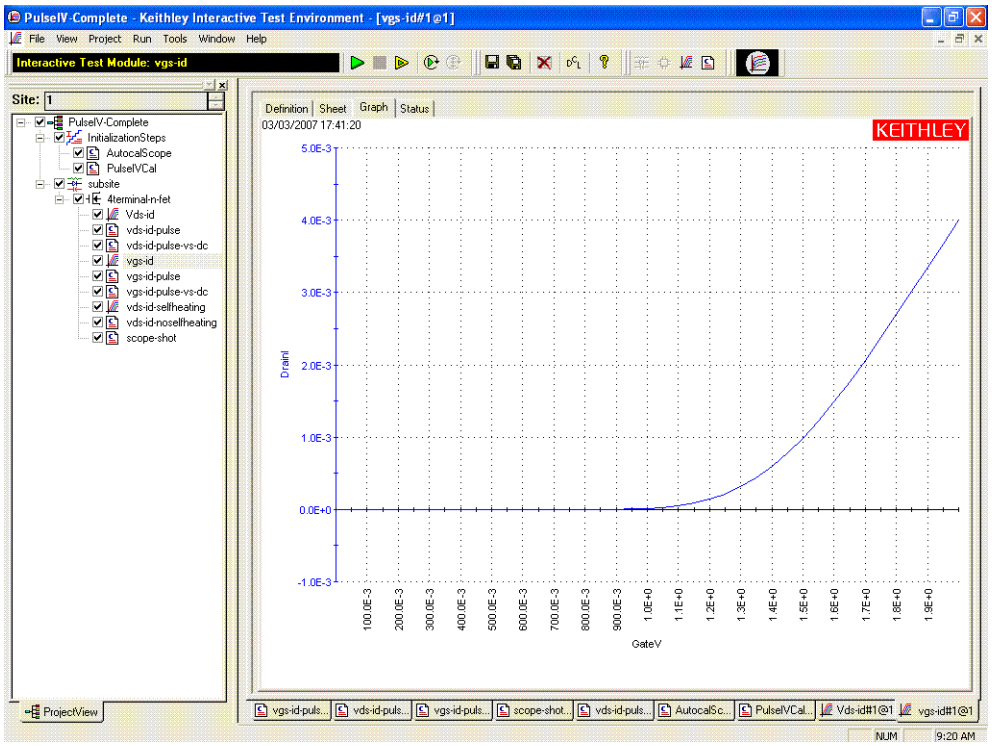


Figure 12-16
DC Vgs-id-pulse UTM Definition tab

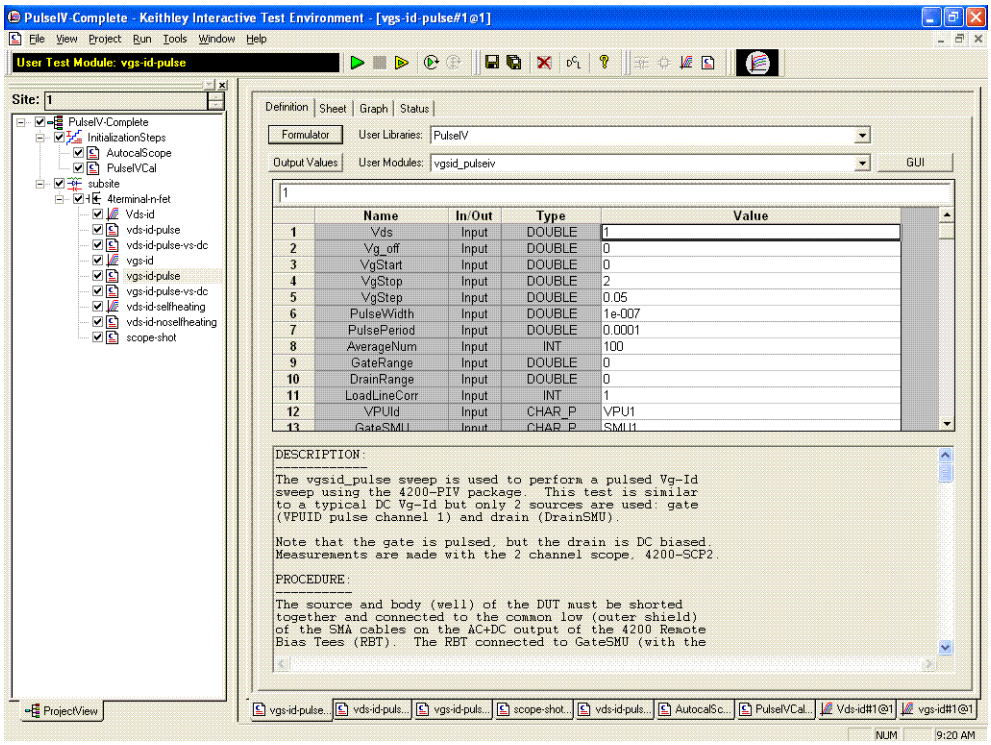


Figure 12-17
DC Vgs-id-pulse UTM GUI

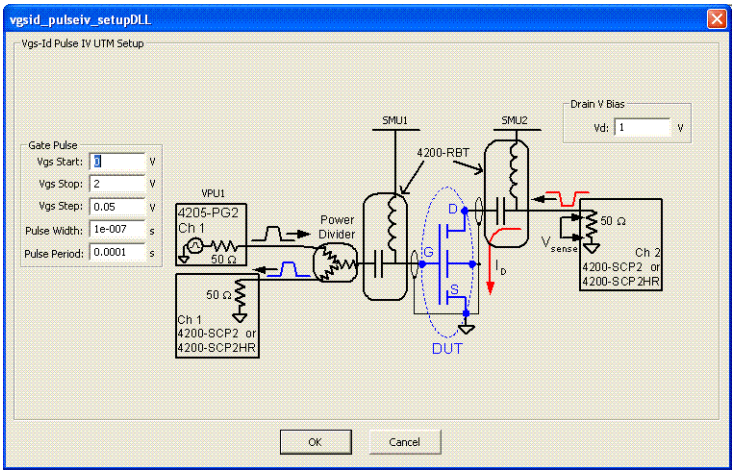
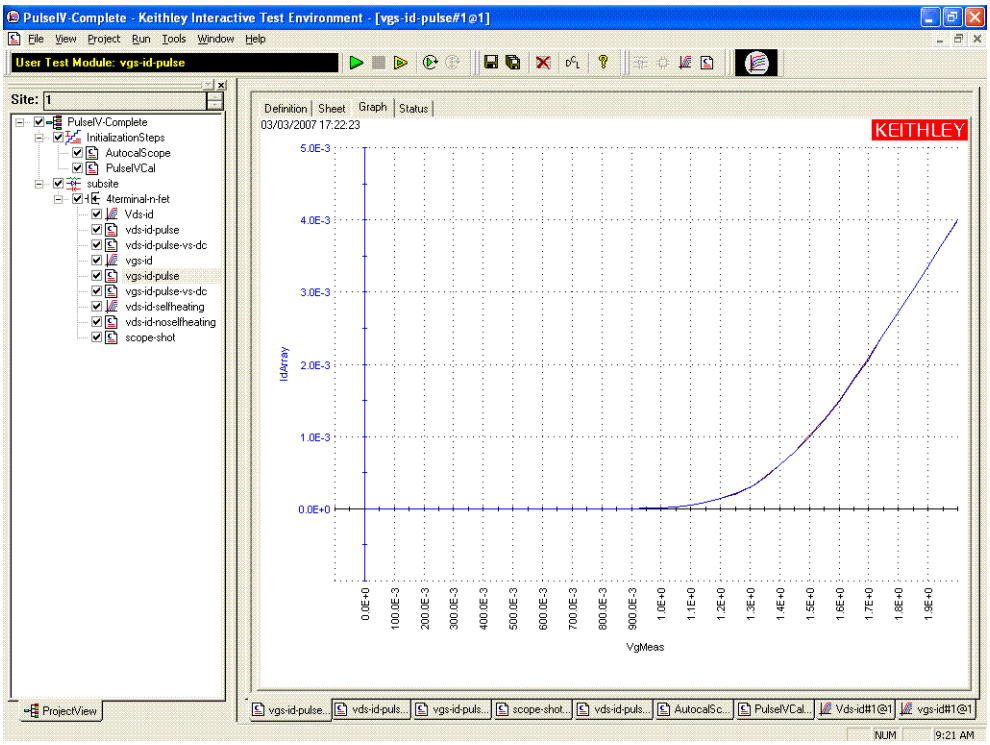


Figure 12-18
Vgs-id-pulse UTM Graph tab



As with Vds-Id, a third vgs-id test (**vgs-id-pulse-vs-dc**) incorporates both pulse and DC into a single UTM. This is useful for comparing DC and pulse results without having to copy data between tests. Figure 12-19 shows the GUI and Figure 12-20 shows the graph. This combined test shares the same Vg and Vd values, in addition to all of the DC and pulse measurement parameters. Note that during the test, neither the graph tab or sheet tab is updated.

Figure 12-19
Pulse and DC Vgs-id UTM GUI

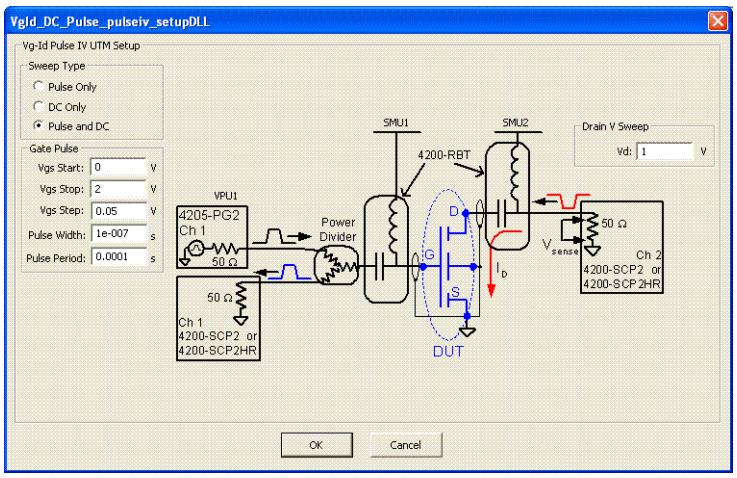
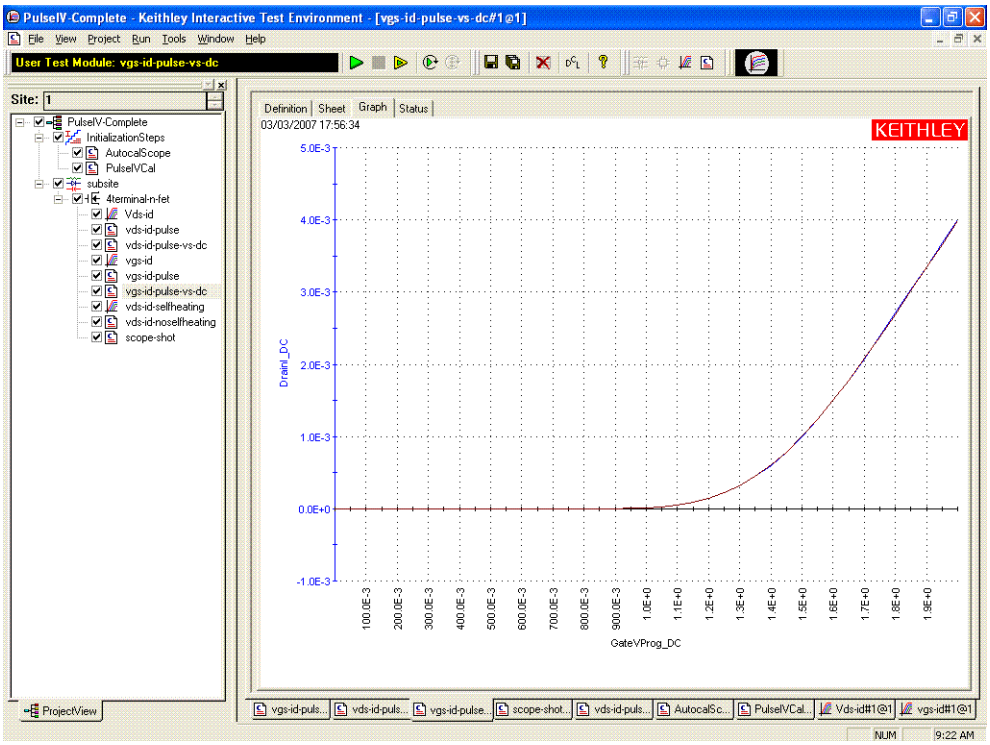


Figure 12-20
Pulse and DC Vgs-id UTM Graph tab

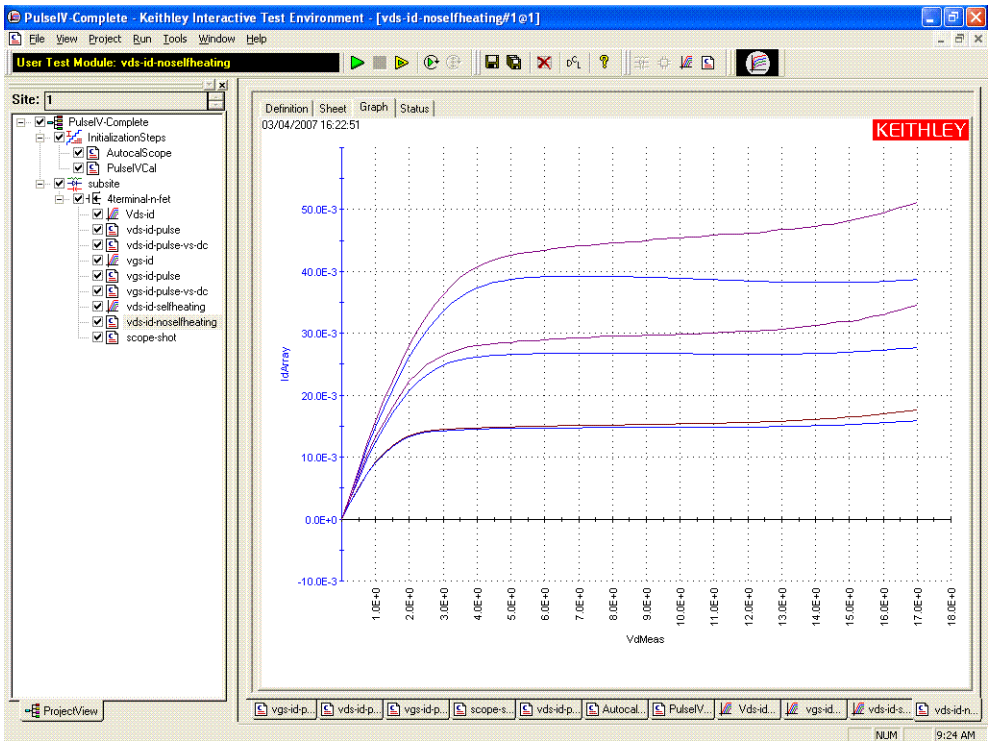


vds-id Self-heating
vds-id No self-heating

This pair of projects is similar to the Vds-id DC and pulse tests described above, but with modifications to the gate and drain voltages to cause self-heating in the DC ITM test results. The pulse test uses the same parameter values, but the low duty cycle (0.1% or less) of the pulses does not heat up the DUT. Figure 12-21 shows the graph of the DC results with corresponding pulse-based Vds-id curves overlaid. Note that the pulse Id values are increasing larger than the corresponding DC curves, indicating a larger heating effect at the higher drain currents.

The purple/red curves are the pulse results in Figure 12-21 corresponding to the DC curves (blue traces).

Figure 12-21
vds-id-noselfheating UTM Graph tab



scope-shot

The scope-shot UTM is a general purpose utility that is used for validation, troubleshooting and for prototyping transient tests. This UTM applies a pulse train to the DUT gate and a DC bias to the DUT drain with a fixed set of values. The returned data is the scope waveform data, as shown in Figure 12-22 and Figure 12-23.

The gate signal is the left pulse curve (blue), using the left Y axis. This curve is the pulse applied to the DUT gate, with approximate calibration values applied. The drain signal is the right pulse (red), using the right Y axis, representing Id, also approximate. Note that the applied drain voltage is DC, but the pulse applied to the gate causes the pulsed response of the drain.

The default graph shows the approximate calibration factors to display the gate voltage and drain current response, although it is possible to display the raw voltages for each scope channel. The sheet tab has the raw voltage from the scope channels, the calibrated measurements for Vg, Vd and Id, as well as the approximate values for the Vg and Id waveforms. In addition, the cursors used to make the pulse measurement are listed. This is useful to determine if the measurement is being made at a flat portion of the pulse. There is no way to provide alternate values for the measurement cursors. Note that the raw waveform data does not have any Pulse IV calibration factors applied, making it useful for a wide range of pulse tests and interconnect configurations.

The graph in Figure 12-22 shows a result for a 100 ns wide pulse. The left pulse curve (blue), using the left Y axis, is the gate voltage applied to the DUT. The right pulse curve (red), using the right Y axis, is the Id response. These values are approximate. The Sheet tab in Figure 12-23 shows the right-most columns of data.

Figure 12-22
scope-shot UTM Graph tab

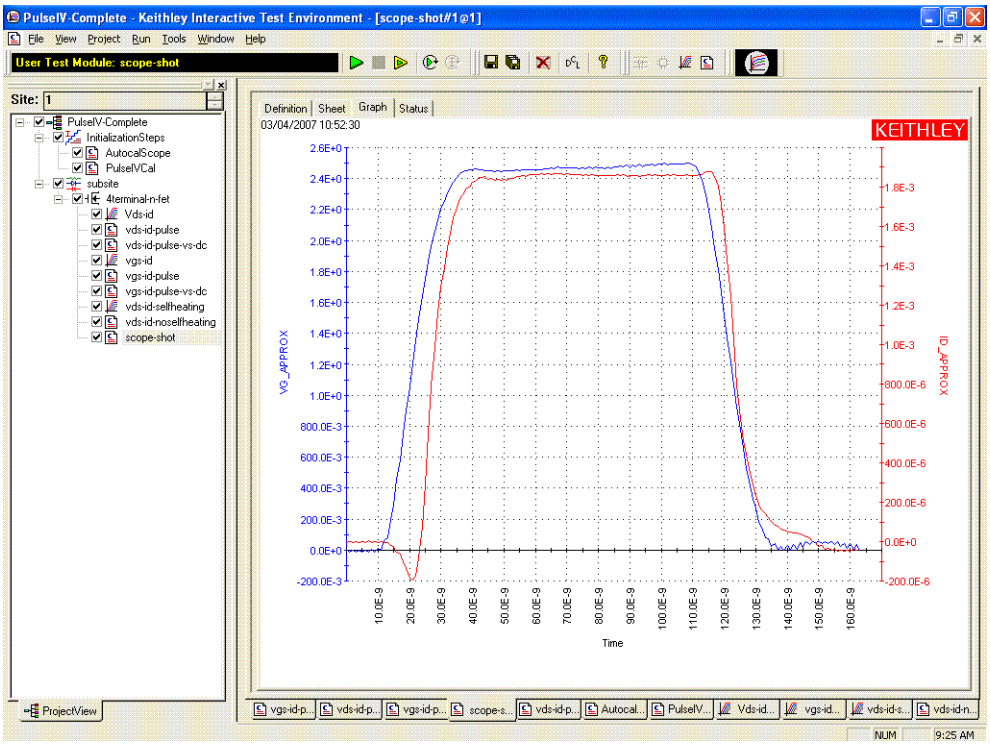
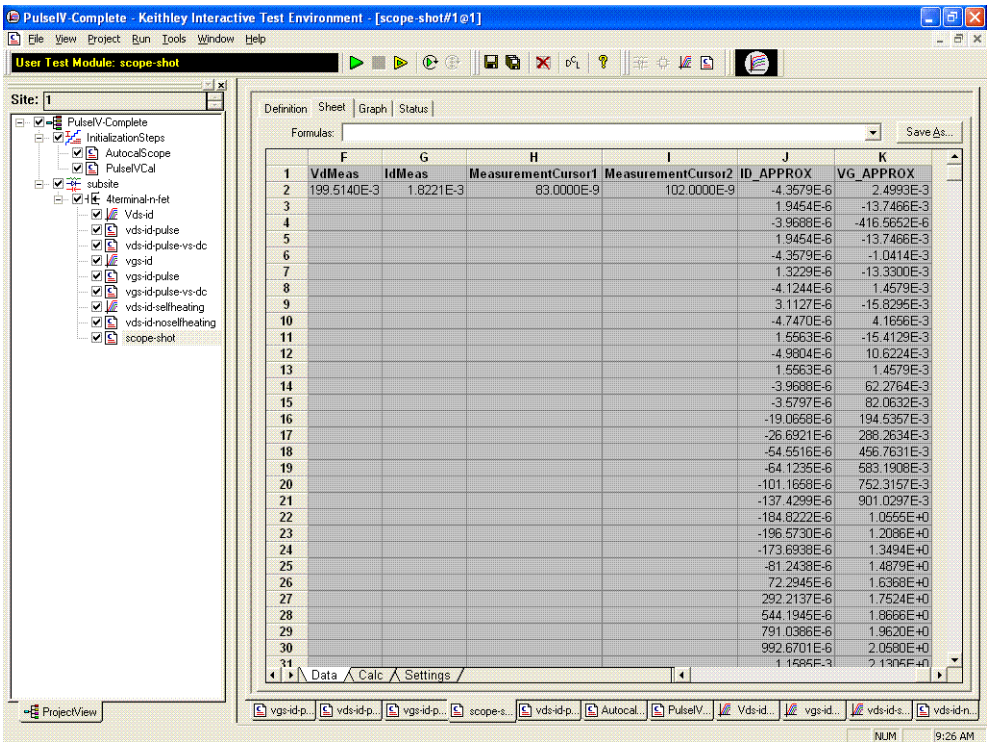


Figure 12-23
scope-shot UTM Sheet tab



QPulseIV-Complete project

In general, the Model 4200-PIV-Q applies pulses to both the gate and the drain of the device under test (DUT). The source and body, if utilized, connections are connected to ground/shield. The dual channel oscilloscope (Model 4200-SCP2HR) measures the gate voltage and current as well as the drain voltage and current.

NOTE *For details on using the QPulseIV-Complete project, see PA-956 (4200-PIV-Q Applications Note).*

Pulse IV UTM descriptions

The pulse IV user library contains modules required to provide low duty cycle pulsed IV testing. The modules contained in the pulse IV user library are listed in Table 12-1 with detailed information following the table.

Table 12-1
Pulse IV UTMs

User Module	Description
cal_pulseiv	Performs a cable compensation routine.
vdsid_pulseiv	Performs a pulsed Vd-Id sweep.
vdId_Pulse_DC_Family_pulseiv	Performs a Pulsed vs. DC Vd-Id sweep.
vgId_DC_DC_pulseiv	Performs a Pulsed vs. DC Vg-Id sweep.
vgsid_pulseiv	Performs a pulsed Vg-Ig sweep.
scopeshot_cal_pulseiv	Used to display a single Pulse IV scopeshot_pulseiv.
scopeshot_pulseiv	Displays a single Pulse IV scopeshot.
vdsid_pulseiv_demo	Performs a pulsed Vd-Id sweep, with simplified parameter list.
vgsid_pulseiv_demo	Performs a pulsed Vg-Id sweep, with simplified parameter list.

cal_pulseiv

Description The cal_pulseiv module is used to perform a cable compensation routine for the 4200-PIV package. This routine permits the system to compensate for losses in the cabling from the 4200 to the connection to the DUT. Use this routine during initial system setup and whenever changes are made in any part of the interconnect (cables, 4200-RBTs, probe manipulators or pins).
There are two main steps to this procedure:

- Open cal—the gate signal is measured while there is no connection to the DUT.
- Through cal—the drain signal is measured while making contact on a Through structure (or by shorting the two 4200-RBTs, AC+DC outputs with an appropriate cable, or adapter).

The factors generated by this routine are used during any testing where the 4200-RBTs are used (vdsid_pulse, vgsid_pulse). Make sure to set the appropriate values for the cal_pulseiv parameters in Table 12-2. Table 12-3 and Table 12-4 contain outputs and return values, respectively.

Connection The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the RBTs. The RBT connected to GateSMU (with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. For detailed connection information, refer to the PIV-A interconnect assembly procedure in Section 3 of the User's Manual.

Table 12-2
Inputs for cal_pulseiv

Input	Type	Description	Default
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with the 4200-PIV package.	VPUID
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SmUs in the system.	GateSMU
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SMUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.	DrainSMU
vRange	int	The pulse generator card voltage source range to be calibrated (V). Valid values are: 5, 20.	5
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 40 us to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%.	100 e-6
Vs_Size Vm1_size Vm2_size	int	Set to a value that is at least equal to the number of steps in the sweep and all three must be the same value.	100 100 100

Table 12-3
Outputs for cal_pulseiv

Output	Type	Description
Vs	double *	The pulse source value (V).
Vm1	double *	The measured voltage from channel 1 of the scope card.
Vm2	double *	The measured voltage from channel 2 of the scope card.

Note: These outputs are included for compatibility with older setups. They no longer return any information.

Table 12-4
Return values for cal_pulseiv

Value	Description
0	OK
-13001	Array Sizes Do Not Match
-13002	Arrays Not Large Enough For Data
-13003	Invalid Instruments
-13004	Unable To Malloc Memory
-13005	Unable To Find Delay Between Channels
-13006	Scope Measurement Error
-13007	Unable To Write To Calibration Files

Table 12-4
Return values for cal_pulseiv

Value	Description
-13008	Invalid Range
-13009	Invalid Calibration Type
-13010	Calibration Data Does Not Meet Correlation Specification
-13998	Calibration Constant Error
-13999	Divider Cal Error

vdsid_pulseiv

Description The vdsid_pulse sweep is used to perform a pulsed Vd-Id sweep using the 4200-PIV package. This test is similar to a typical DC Vd-Id but only two sources are used: gate (VPUID pulse channel 1) and drain (DrainSMU). The gate is pulsed, but the drain is DC biased.

Measurements are made with the two channel scope card. To create a family of Vd-Id curves, change Vgs and run the test by using the append button. Make sure to set the appropriate values for the Vds-Id parameters (see Table 12-5). Table 12-6 and Table 12-7 contain outputs and return values, respectively.

Connection The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the 4200-RBT. The RBT connected to GateSMU (with the power divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. For detailed connection information, refer to the PIV-A interconnect assembly procedure in Section 3 of the User's Manual..

Table 12-5
Inputs for vdsid_pulseiv

Input	Type	Description
Vgs	double	The pulsed gate-source voltage bias, output by channel 1 of the pulse generator card (VPUID).
Vg_off	double	The DC bias applied by the GateSMU to put device in the OFF state. Normally set to 0 V for enhancement FETs (may be non-zero for depletion FETs).
VdStart	double	The starting sweep value for Vd, output by the DrainSMU (defined below).
VdStop	double	The final sweep value for Vd, output by the DrainSMU (defined below).
VdStep	double	The sweep step size for the Vd sweep, output by the DrainSMU (defined below).
PulseWidth	double	The Vgs pulse width (PW). The PW can be 40 ns to 150 ns (10 ns resolution). Pulses wider than 150 ns will begin to be attenuated by the capacitor in the 4200-RBT.
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 100µs to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. The period is most easily calculated by multiplying the largest desired pulse width (PW) by1000. Example: PW = 150 ns, so Period = 150 us.
AverageNum	int	The number of pulses to average at each step of the sweep. For best low signal performance, set AverageNum = 0 for Adaptive Filtering.

Table 12-5 (cont.)
Inputs for vdsid_pulseiv

Input	Type	Description
GateRange	double	The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10.
DrainRange	double	The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range, where V = I * 50 Ω. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10.
LoadLineCorr	int	Determines whether to use load line correction to compensate for the voltage drop caused by the 50 Ω sense resistor used to measure the drain current (Id). 1 = load line correction active. 0 = no load line correction.
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with the 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SmUs in the system.
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SMUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.
IdSize VdMeasSize VdProgSize VgSize VgProgSize	int	Set to a value that is at least equal to the number of steps in the sweep and all five must be the same value.

Table 12-6
Outputs for vdsid_pulseiv

Output	Type	Description
IdArray	double *	The measured drain current from channel 2 of the scope card. This current is determined by measuring the voltage drop across the scope card 50 Ω termination, giving Id = Vd / 50 Ω.
VdMeas	double *	Array of measured drain voltage values.
VdProg	double *	Array of programmed drain voltage values.
VgMeas	double *	The measured gate voltage from channel 1 of the scope card.
VgProg	double *	Array of programmed gate voltage values.

Table 12-7
Return values for vdsid_pulseiv

Value	Description
0	OK
-1	Invalid value for Vgs
-2	Invalid value for VdStart
-3	Invalid value for VdStop
-4	Invalid value for VdStep
-5	Invalid value for PulseWidth
-6	Invalid value for PulsePeriod

Table 12-7
Return values for vdsid_pulseiv

Value	Description
-7	Invalid value for AverageNum
-8	Invalid value for LoadLineCorr
-9	Array sizes do not match
-10	Array sizes not large enough for sweep
-11	Invalid VPUId
-12	Invalid GateSMU
-13	Invalid DrainSMU
-14	Unable to initialize PIV solution

VdId_Pulse_DC_Family_pulseiv

Description

The VdId_Pulse_DC_Family_pulseiv sweep is used to perform a Pulsed vs DC Vd-Id sweep using the 4200-PIV-A package. This test is similar to a typical Vd-Id but only two sources are used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2-channel scope, 4200-SCP2.
To create a family of curves, choose an appropriate start and stop value for Vgs, and a number of steps.

This routine can run the sweeps in three different ways: 1) DC only; 2) Pulse only; 3) Pulse and DC curves. This routine supports from one to 10 Vd-Id curves based on up to 10 different Vgs values.

This routine also supports the 4200-PIV-A package using the 4200-RBT. For this package, all test parameters and limits are given below, except the 4200-PIV-A with the 4200-RBT has a max pulse width of 150 ns, not the 250 ns of the 4205-RBT.

All voltage levels specified below assume a 50 Ω DUT load.

Connection

The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the 4205-RBT. The RBT connected to GateSMU (the RBT with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. Use either G-S-G probes for RF structures, or use DC probes with the 4200-PRB-C adapter cables for DC structures.

Set the appropriate values for the Vds-Id parameters. Inputs, outputs and returned values are provided in Table 12-8, Table 12-9 and Table 12-10.

Table 12-8
Inputs for VdId_Pulse_DC_Family_pulseiv

Input	Type	Description
VgStart	double	The starting step value for Vg. For DC only sweeps, VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V.

Table 12-8 (cont.)
Inputs for VdId_Pulse_DC_Family_pulseiv

Input	Type	Description
VgStop	double	The final step value for Vg. For DC only sweeps, VgStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStop must be between -5 V to +5 V.
VgNumSteps	double	The number of steps for Vg (Max = 10).
Vg_off	double	The DC bias applied by the GateSMU to put device in the OFF state. Set to 0 V for enhancement FETs (may be non-zero for depletion FETs).
VdStart	double	The starting sweep value for Vd. For DC only sweeps, VdStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VdStop	double	The final sweep value for Vd. For DC only sweeps, VdStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT.
VdStep	double	The number of steps for the Vd sweep (Max = 10000).
PulseWidth	double	The Vgs pulse width (PW). The PW can be 40 ns to 250 ns (10 ns resolution). Pulses wider than 250 ns will begin to be attenuated by the coupling capacitor in the Remote Bias Tee (4205-RBT).
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 100µs to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. The period is most easily calculated by multiplying the largest desired pulse width (PW) by1000. Example: PW = 150 ns, so Period = 150µs.
AverageNum	int	The number of pulses to average at each step of the sweep. For best low signal performance, set AverageNum = 0 for Adaptive Filtering.
GateScpRange	double	The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10. These ranges are Vpp. For example, the 0.5 range covers -250 to +250 mV.
DrainScpRange	double	The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range, where V = I * 50 Ω. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10. These ranges are Vpp. For example, the 0.5 range covers -250 to +250 mV.

Table 12-8 (cont.)
Inputs for VdId_Pulse_DC_Family_pulseiv

Input	Type	Description
GateSMURange	int	The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
DrainSMURange	int	The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
LoadLineCorr	int	Determines whether to use load line correction to compensate for the voltage drop caused by the 50 Ω sense resistor used to measure the drain current (Id). 1 = load line correction active. 0 = no load line correction.
GateCompliance	double	The SMU current compliance for the DUT Gate.
DrainCompliance	double	The SMU current compliance for the DUT Drain.
NPLC	double	The DC measurement integration time in NPLC (Number of Power Line cycles).
DCSourceDelay	double	Time, in seconds, between the DC source and measure for each sweep point.
DC_vs_Pulse	int	Determines whether to run a DC and Pulse test or a DC only test or a Pulse only test. 0 - Pulse Only, 1 - DC Only, 2 - DC and Pulse.
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with a 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SMUs in the system.

Table 12-8 (cont.)
Inputs for VdId_Pulse_DC_Family_pulseiv

Input	Type	Description
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SMUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.
DrainVMeas_DC_Size DrainVProg_DC_Size DrainI_DC_Size GateVMeas_DC_Size GateVProg_DC_Size DrainVMeas_Pulse_Size DrainVProg_Pulse_Size DrainI_Pulse_Size GateVMeas_Pulse_Size GateIProg_Pulse_Size	int	Sizes of the output arrays. All arrays should be the same size and need to be large enough to hold all sweep points.

Table 12-9
Outputs for Vdid_Pulse_DC_Family_pulseiv

Output	Type	Description
DrainVProg_DC DrainVProg_Pulse	double	Array of programmed drain voltage values.
DrainVMeas_DC DrainVMeas_Pulse	double	Array of measured drain voltage values.
DrainI_DC DrainI_Pulse	double	Array of measured drain currents.
GateVMeas_DC GateVMeas_Pulse	double	Array of measured gate voltages.
GateVProg_DC GateVProg_Pulse	double	Array of programmed gate voltages.

Table 12-10
Return values for Vdid_Pulse_DC_Family_pulseiv

Value	Description
0	OK
-1	Invalid value for Vgs
-2	Invalid value for VdStart
-3	Invalid value for VdStop
-4	Invalid value for VdStep
-5	Invalid value for PulseWidth
-6	Invalid value for PulsePeriod
-7	Invalid value for AverageNum

Table 12-10
Return values for Vdid_Pulse_DC_Family_pulseiv

Value	Description
-8	Invalid value for LoadLineCorr
-9	Array sizes do not match
-10	Array sizes not large enough for sweep
-11	Invalid VPUId
-12	Invalid GateSMU
-13	Invalid DrainSMU
-14	Unable to initialize PIV solution
-15	Invalid GateSMU Range
-16	Invalid DrainSMU Range

vgsid_pulseiv

Description The vgsid_pulse sweep is used to perform a pulsed Vg-Ig sweep using the 4200-PIV package. This test is similar to a typical DC Vg-Id but only two sources are used: gate (VPUID pulse channel 1) and drain (DrainSMU). The gate is pulsed, but the drain is DC biased.
Measurements are made with the 2 channel scope card. Set the appropriate values for the Vgs-Id parameters (Table 12-11). Table 12-12 and Table 12-13 contain outputs and return values, respectively.

Connection The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the 4200-RBT. The RBT connected to GateSMU (with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. For detailed connection information, refer to the PIV-A interconnect assembly procedure in Section 3 of the User's Manual.

Table 12-11
Inputs for vgsid_pulseiv

Input	Type	Description
Vds	double	The drain-source voltage, output by the DrainSMU (defined below).
Vg_off	double	The DC bias applied by the GateSMU to put device in the OFF state. Normally set to 0 V for enhancement FETs (may be non-zero for depletion FETs).
VgStart	double	The starting sweep value for Vg, output by channel 1 of the pulse generator card (VPUID).
VgStop	double	The final sweep value for Vg, output by channel 1 of the pulse generator card (VPUID).
VgStep	double	The sweep step size for the Vg sweep, output by channel 1 of the pulse generator card (VPUID).
PulseWidth	double	The Vgs pulse width (PW). The PW can be 40 ns to 150 ns (10 ns resolution). Pulses wider than 150 ns will begin to be attenuated by the capacitor in the 4200-RBT.

Table 12-11
Inputs for vgsid_pulseiv

Input	Type	Description
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 100µs to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. The period is most easily calculated by multiplying the largest desired pulse width (PW) by1000. Example: PW = 150 ns, so Period = 150 us.
AverageNum	int	The number of pulses to average at each step of the sweep. For best low signal performance, set AverageNum = 0 for Adaptive Filtering.
GateRange	double	The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10.
DrainRange	double	The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range, where V = I * 50 Ω. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10.
LoadLineCorr	int	Determines whether to use load line correction to compensate for the voltage drop caused by the 50 Ω sense resistor used to measure the drain current (Id). 1 = load line correction active. 0 = no load line correction.
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with the 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SmUs in the system.
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SmUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.
IdSize VgMeasSize VgProgSize VdMeasSize VdProgSize	int	Set to a value that is at least equal to the number of steps in the sweep and all five must be the same value.

Table 12-12
Outputs for Vgsid_pulseiv

Output	Type	Description
VgMeas	double	Array of measured gate voltage values.
VgProg	double	Array of programmed gate voltage values.
VdMeas	double	Array of measured drain currents.
VdProg	double	Array of programmed drain voltages.
IdArray	double	Array of measured drain current values.

Table 12-13
Return values for vgsid_pulseiv

Value	Description
0	OK
-1	Invalid value for Vds
-2	Invalid value for VgStart
-3	Invalid value for VgStop
-4	Invalid value for VgStep
-5	Invalid value for PulseWidth
-6	Invalid value for PulsePeriod
-7	Invalid value for AverageNum
-8	Invalid value for LoadLineCorr
-9	Array sizes do not match
-10	Array sizes not large enough for sweep
-11	Invalid VPUId
-12	Invalid GateSMU
-13	Invalid DrainSMU
-14	Unable to initialize PIV solution

VgId_DC_Pulse_pulseiv

Description

The VgId_DC _Pulse pulseiv sweep is used to perform a Pulsed vs DC Vg-Id sweep using the 4200-PIV-A package. This test is similar to a typical Vg-Id but only two sources are used: one for the DUT Gate and one for the DUT Drain. Pulsed Measurements are made with the 2-channel scope, 4200-SCP2.

This routine can run the sweeps in three different ways: 1) DC only; 2) Pulse only; 3) Pulse and DC curves. This routine supports from one to 10 Vd-Id curves based on up to 10 different Vgs values.

This routine also supports the 4200-PIV-A package using the 4200-RBT. For this package, all test parameters and limits are given below, except the 4200-PIV-A with the 4200-RBT has a max pulse width of 150 ns, not the 250 ns of the 4205-RBT.

All voltage levels specified below assume a 50 Ω DUT load.

Connection

The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the 4205-RBT. The RBT connected to GateSMU (the RBT with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. Use either G-S-G probes for RF structures, or use DC probes with the 4200-PRB-C adapter cables for DC structures.

Set the appropriate values for the Vds-Id parameters. Inputs, outputs and returned values are provided in Table 12-14, Table 12-15 and Table 12-16.

Table 12-14
Inputs for Vgid_DC_Pulse_pulseiv

Input	Type	Description
Vds	double	The voltage value for Vd. For DC only sweeps, Vds must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, Vds must be between -5 V to +5 V.
VgStart	double	The starting step value for Vg. For DC only sweeps, VgStart must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStart must be between -5 V to +5 V.
VgStop	double	The final step value for Vg. For DC only sweeps, VgStop must be between -200 V to +200 V dependent on the type of SMU and the current requirements of the DUT. For pulse and pulse and DC Sweeps, VgStop must be between -5 V to +5 V.
VgStep	double	The sweep step size for the Vg sweep, output by channel 1 of the 4200-PG2 (VPUID).
Vg_off	double	The DC bias applied by the GateSMU to put device in the OFF state. Normally set to 0 V for enhancement FETs (may be non-zero for depletion FETs). This package does not support a similar capability for the drain. For full quiescent, or bias, point testing, review the 4200-PIV-Q specs.
PulseWidth	double	The Vgs pulse width (PW). The PW can be 40 ns to 250 ns (10 ns resolution). Pulses wider than 250 ns will begin to be attenuated by the coupling capacitor in the Remote Bias Tee (4205-RBT).
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 100 µs to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. The period is most easily calculated by multiplying the largest desired pulse width (PW) by 1000. Example: PW = 150 ns, so Period = 150 µs.
AverageNum	int	The number of pulses to average at each step of the sweep. For best low signal performance, set AverageNum = 0 for Adaptive Filtering.
GateScpRange	double	The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10. These ranges are Vpp. For example, the 0.5 range covers -250 to +250 mV.
DrainScpRange	double	The voltage measure range for the scope channel measuring the Drain. Use 0 for scope autoranging, or specify a voltage value for a fixed range, where V = I * 50 Ω. Valid voltages are 0.050, 0.1, 0.2, 0.5, 1, 2, 5, 10. These ranges are Vpp. For example, the 0.5 range covers -250 to +250 mV.

Table 12-14 (cont.)
Inputs for Vgid_DC_Pulse_pulseiv

Input	Type	Description
GateSMURange	int	The current measurement range to be used for the SMU on the DUT Gate terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
DrainSMURange	int	The current measurement range to be used for the SMU on the DUT Drain terminal. Values correspond to the table below. Limited Auto means that the value given is the minimum measurement range used, with automatic ranging for larger currents. 1 Full Auto 2 Limited Auto 10 pA 3 Limited Auto 100 pA 4 Limited Auto 1 nA 5 Limited Auto 10 nA 6 Limited Auto 100 nA 7 Limited Auto 1 uA 8 Limited Auto 10 uA 9 Limited Auto 100 uA 10 Limited Auto 1 mA 11 Limited Auto 10 mA 12 Limited Auto 100 mA
LoadLineCorr	int	Determines whether to use load line correction to compensate for the voltage drop caused by the 50 Ω sense resistor used to measure the drain current (Id). 1 = load line correction active. 0 = no load line correction.
GateCompliance	double	The SMU current compliance for the DUT Gate.
DrainCompliance	double	The SMU current compliance for the DUT Drain.
NPLC	double	The DC measurement integration time in NPLC (Number of Power Line cycles).
DCSourceDelay	double	Time, in seconds, between the DC source and measure for each sweep point.
DC_vs_Pulse	int	Determines whether to run a DC and Pulse test or a DC only test or a Pulse only test. 0 - Pulse Only, 1 - DC Only, 2 - DC and Pulse.
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with a 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SMUs in the system.

Table 12-14 (cont.)
Inputs for Vgid_DC_Pulse_pulseiv

Input	Type	Description
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SMUs in the system. This is the SMU that applies the DC bias to the DUT drain during the pulse or DC sweep.
Draini_DC_X_Size DrainVMeas_DC_Size DrainVProg_DC_Size GateVMeas_DC_X GateVProg_DC_Size DrainISize_Pulse_Size DrainVMeas_Pulse_Size DrainVProg_Pulse_Size GateVSize_Pulse_Size GateVProg_Pulse_Size	int	These values *must* be set to a value that is at least equal to the number of steps in the sweep and all values must be the same.

Table 12-15
Outputs for Vgid_DC_Pulse_pulseiv

Output	Type	Description
DrainI_DC/Pulse	double	The measured drain current from channel 2 of the 4200-SCP2 or the DrainSMU. In the case of Pulse, this current is determined by measuring the voltage drop across the 4200-SCP2 50 Ω termination, giving Id = Vd / 50 Ω.
DrainVMeas_DC/Pulse	double	The measured drain voltage from channel 2 of the 4200-SCP2 in the case of pulse and the measured voltage on the DrainSMU in the case of DC.
DrainV_Prog_DC/Pulse	double	The programmed drain voltage, either supplied by the PG2 or the SMU for the drain.
GateVMeas_DC/Pulse	double	The measure gate voltage from channel 1 of the 4200-SCP2 in the case of pulse and the measured voltage on the GateSMU in the case of DC.
GateVProg_DC/Pulse	double	The programmed gate voltage, either supplied by the PG2 or Gate SMU.

Table 12-16
Return values for Vgid_DC_Pulse_pulseiv

Value	Description
0	OK
-1	Invalid value for Vds
-2	Invalid value for VgStart
-3	Invalid value for VgStop
-4	Invalid value for VgStep
-5	Invalid value for PulseWidth
-6	Invalid value for PulsePeriod

Table 12-16
Return values for Vgid_DC_Pulse_pulseiv

Value	Description
-7	Invalid value for AverageNum
-8	Invalid value for LoadLineCorr
-9	Array sizes do not match
-10	Array sizes not large enough for sweep
-11	Invalid VPUId
-12	Invalid GateSMU
-13	Invalid DrainSMU
-14	Unable to initialize PIV solution
-15	Invalid GateSMU Range
-16	Invalid DrainSMU Range

scopeshot_cal_pulseiv

Description The scopeshot_cal_pulseiv routine is used to display a single Pulse IV scopeshot_pulseiv. This routine is useful to understand the basic source and measure concepts behind the Pulse IV methods for pulse vds-id and vgs-id. Measurements are made with cable compensation values applied to them and load line compensation can be used if desired.

Connection The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the RBT. The RBT connected to GateSMU (with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain.

Set the appropriate values for the scopeshot_cal_pulseiv. Set the appropriate values for the Vds-Id parameters. Inputs, outputs and returned values are provided in Table 12-17, Table 12-18 and Table 12-19.

Table 12-17
Inputs for scopeshot_cal_pulseiv

Input	Type	Description
Vds	double	The DC drain bias, provided by the DrainSMU.
Vgs	double	The pulse gate voltage amplitude. This can be set from -5 to +5 V.
VgStart	double	The starting sweep value for Vg, output by channel 1 of the pulse generator card (VPUID).
PulseWidth	double	The Vgs pulse width (PW). The PW can be 40 ns to 300 ns (10 ns resolution).
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 100 us to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. This period is most easily calculated by multiplying the largest desired pulse width (PW) by 1000. Example: PW = 150 ns, so Period = 150 us.
AverageNum	int	The number of pulses to average at each step of the sweep. For best low signal performance, set AverageNum = 0 for Adaptive Filtering.

Table 12-17
Inputs for scopeshot_cal_pulseiv

Input	Type	Description
GateRange	double	The voltage measure range for the scope channel measuring the Gate. Use 0 for scope autoranging, or specify a voltage value for a fixed range. Valid voltages are 0.25, 0.5, 1.25, 2.5, 5, 10, 25, 50. The range is a full range value (for example, 2.5 is -1.25 V to +1.25 V).
LoadLineCorr	int	Determines whether to use load line correction to compensate for the voltage drop caused by the DUT impedance on the Drain. When load line correction is on (1), the test will start by assuming a high impedance value for the device and will approach the correct bias and pulse values over a series of pulses, that ensures that the sourced pulses match the requested values. When load line correction is turned off, the specified voltages will be sourced (1 = Use Load Line, 0 = No Load Line).
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with the 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SmUs in the system.
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SmUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.
TimeSize GatePulseSize DrainPulseSize	int	These values must be set to a GatePulseSize value that is at least equal to the DrainPulseSize number of steps in the sweep and all three must be the same value.

Table 12-18
Outputs for scopeshot_cal_pulseiv

Output	Type	Description
Time	double *	Array of time values from the 4200-SCP2 scope (s).
GatePulse	double *	Array of gate pulse voltages from channel 1 of the 4200-SCP2 scope.
DrainPulse	double *	Array of drain voltages from channel 2 of the 4200-SCP2 scope.
VgMeas	double *	Measured Gate Voltage.
VdMeas	double *	Measured Drain Voltage.
IdMeas	double *	Measured Gate Current.

Table 12-19
Return values for scopeshot_cal_pulseiv

Value	Description
0	OK
-1	Invalid Gate Voltage (Max 5 V).
-2	Invalid Drain Voltage (Max 210 V).
-5	Invalid Pulse Width (Min 40 ns).
-6	Invalid Pulse Period (Min 40 ns).
-7	Invalid Average Num (must be between 1 and 100000).

Table 12-19
Return values for scopeshot_cal_pulseiv

Value	Description
-8	Invalid LoadLineCorr (must be between 0 and 1).
-9	Time, GatePulse, and Drain Pulse array sizes must be equal.
-11	Invalid VPU. Specified VPU is not in current system configuration.
-12	Invalid GateSMU. Specified SMU is not in current system configuration.
-13	Invalid DrainSMU. Specified SMU is not in current system configuration.
-14	PIV Initialization Failed.

scopeshot_pulseiv

Description The scopeshot_pulseiv routine displays a single Pulse IV scopeshot. This routine is useful to understand the basic source and measure concepts behind the Pulse IV methods for pulse vds-id and vgs-id. The scope waveforms are retrieved and displayed for both channels (no measurements are made). Make sure to set the appropriate values for the scopeshot_pulseiv (see Table 12-20). Table 12-21 and Table 12-22 contain outputs and return values, respectively.

Connection The source and body (well) of the DUT must be shorted together and connected to the common low (outer shield) of the SMA cables on the AC+DC output of the 4200-RBT. The RBT connected to GateSMU (with the Power Divider) should be connected to the gate. The RBT connected to DrainSMU should be connected to the drain. For detailed connection information, refer to the PIV-A interconnect assembly procedure in Section 3 of the User's Manual..

Table 12-20
Inputs for scopeshot_pulseiv

Input	Type	Description
RiseTime	double	The gate pulse transition rise time (s). This can be set from 10 e-9 to 300 e-9 in 10 e-9 (10 ns) steps. This value programs the full transition time (0–100%), not the 10–90% time.
FallTime	double	The gate pulse transition fall time (s). This can be set from 10 e-9 to 300 e-9 in 10 e-9 (10 ns) steps. This value programs the full transition time (0–100%), not the 10–90% time.
PulseWidth	double	The gate pulse width (PW). The PW can be 20 ns to 1us (10 ns resolution). Pulses wider than 150 ns will begin to be attenuated by the capacitor in the 4200-RBT.
PulseBase	double	The pulse gate base voltage. This can be set from -5 to +5 V, inclusive of amplitude.
PulseAmplitude	double	The pulse gate voltage amplitude. This can be set from -5 to +5 V, inclusive of base voltage.
GateLoad	double	The scope card channel 1 input impedance for the gate. Either 50 or 1E6. Use 50 for Pulse IV with RBTs.
GateRange	double	The scope card channel 1 Y scale voltage range for the gate measurement. Typical values are 1, 2, 5 V.
DrainLoad	double	The scope card channel 2 input impedance for the drain. Either 50 or 1E6. Use 50 for Pulse IV with RBTs.
DrainRange	double	The scope card channel 2 Y scale voltage range for the drain measurement. Typical values are 1, 2, 5 V.
PulsePeriod	double	The pulse period for the Vgs pulse. The period can be set from 40 ns to 1 s (10 ns resolution). The period must be set so that the Duty Cycle (DC) is no more than 0.1%. This period is most easily calculated by multiplying the largest desired pulse width (PW) by 1000. Example: PW = 150 ns, so Period = 150 us.
AverageNum	int	The number of waveforms to average.
GateBias	double	The DC gate bias, provided by the gateSMU.
DrainBias	double	The DC drain bias, provided by the drainSMU.
VPUID	char *	The instrument ID. This should be set to VPU1 for 4200 systems with the 4200-PIV package.
GateSMU	char *	The SMU used for the Gate. This can be SMU1 up to the maximum number of SmUs in the system.
DrainSMU	char *	The SMU used for the Drain. This can be SMU1 up to the maximum number of SMUs in the system. This is the SMU that applies the DC bias to the DUT drain during the sweep.
TimeSize Ch1OutSize Ch2OutSize	int	Set to a value that is at least equal to the number of steps in the sweep and all three must be the same value.

Table 12-21
Outputs for scopeshot_pulseiv

Output	Type	Description
Time	double *	Array of time values from the scope card (s).
Ch1Out	double *	Array of gate voltages from channel 1 of the scope card.
Ch2Out	double *	Array of drain voltages from channel 2 of the scope card.

Table 12-22
Return values for scopeshot_pulseiv

Value	Description
0	OK
-1	Invalid Pulse Width (Min 40 ns)
-2	Invalid Pulse Period (Min 40 ns)
-3	Invalid Average Num (1 - 1000)
-4	Array Sizes Do Not Match
-5	Invalid VPU. Specified VPU Is Not In Current System Configuration
-6	Invalid GateSMU. Specified SMU Is Not In Current System Configuration
-7	Invalid DrainSMU. Specified SMU Is Not In Current System Configuration
Negative numbers are errors—refer to LPT and PulseIV documentation for description.	

vdsid_pulseiv_demo

(See vdsid_pulseiv in Section 3 of the User's Manual)

vgsid_pulseiv_demo

(See vgsid_pulseiv in Section 3 of the User's Manual)

scopeshot_pulseiv_demo

(See scopeshot_pulseiv in Section 3 of the User's Manual)

NOTE *The above three UTMs are functionally identical but simpler than their respective routines listed earlier in this section of the manual. The difference being less-used parameters have been eliminated from the parameter list and hard-coded (for example, SMU channels, ranges, load line).*

Section 13

KPulse (for Keithley Pulse Generator Cards)

NOTE *The existing Model 4200-SCS Reference Manual Section 13, KPulse, has been removed from the Reference Manual and moved to the Model 4200 User's Manual, How to Generate Basic Pulses, Section 5*

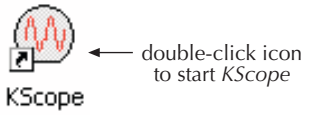
Section 14

KScope (for Models 4200-SCP2 and 4200-SCP2HR)

In this section:

KScope graphical user interface

KScope is a graphical user interface (GUI) provided with the Keithley Instruments Model 4200-SCS Semiconductor Characterization System® that provides a non-programming alternative to control the system's scope card (either Model 4200-SCP2HR or Model 4200SCP2). The GUI, which is shown in Figure 14-1, can be opened by double-clicking the KScope icon on the desktop:



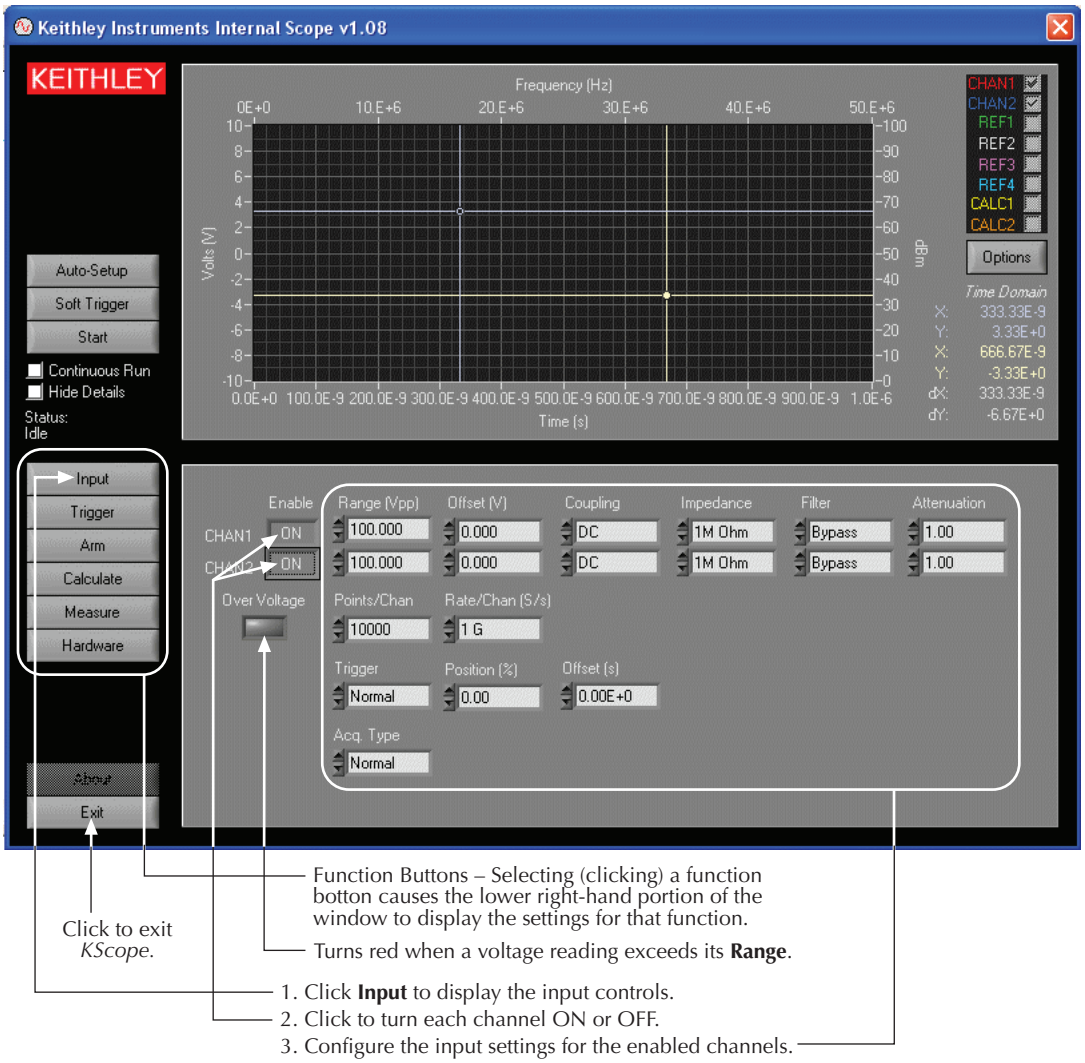
The following figures contain explanations of how to configure and operate the scope GUI. See the Model 4200-SCS User's Manual, Scope card settings, for information on the scope settings. Complete details on scope settings are provided in the ZTEC User's Manual.

- Figure 14-1 explains how to Configure Input settings for the scope
- Figure 14-2 explains how to Configure Trigger settings for the scope
- Figure 14-3 explains how to Configure Arm settings for the scope
- Figure 14-4 explains how to Configure Calculate settings for the scope
- Figure 14-5 explains how to Configure Measure settings for the scope
- Figure 14-6 explains how to Configure the Hardware settings for the scope
- Figure 14-7 explains how to Operate the scope

Configure Input settings

Figure 14-1 explains how to configure the **Input** settings for the scope.

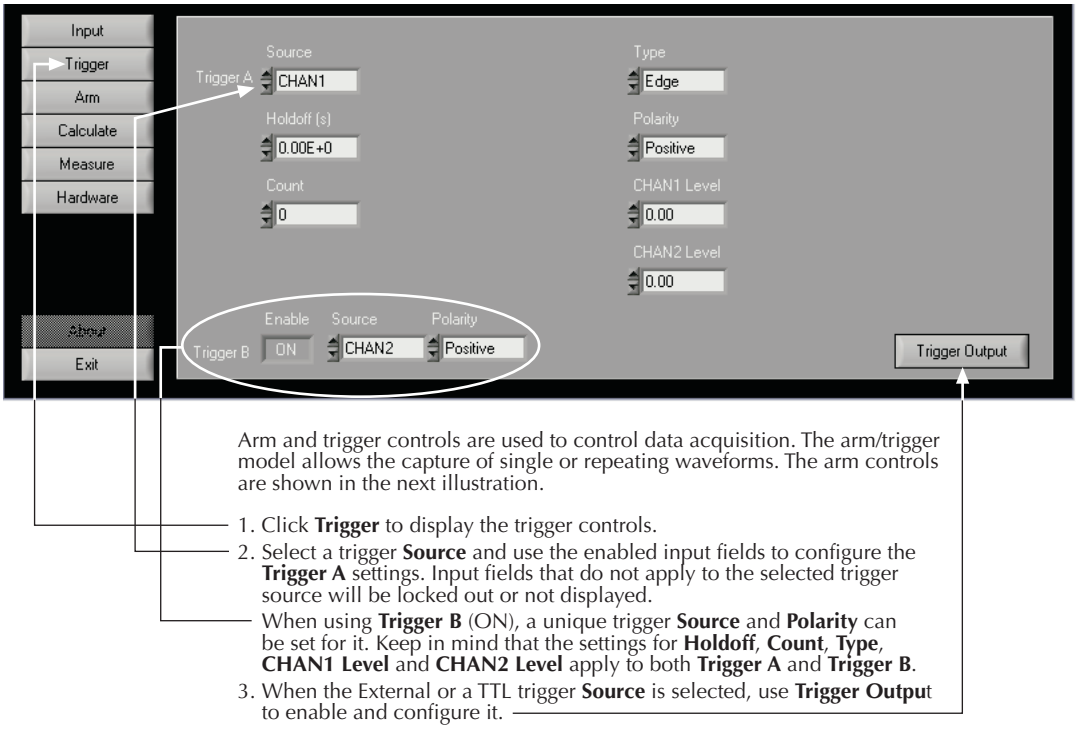
Figure 14-1
KScope: Configuring the Input settings



Configure Trigger settings

Figure 14-2 explains how to configure the **Trigger** settings for the scope.

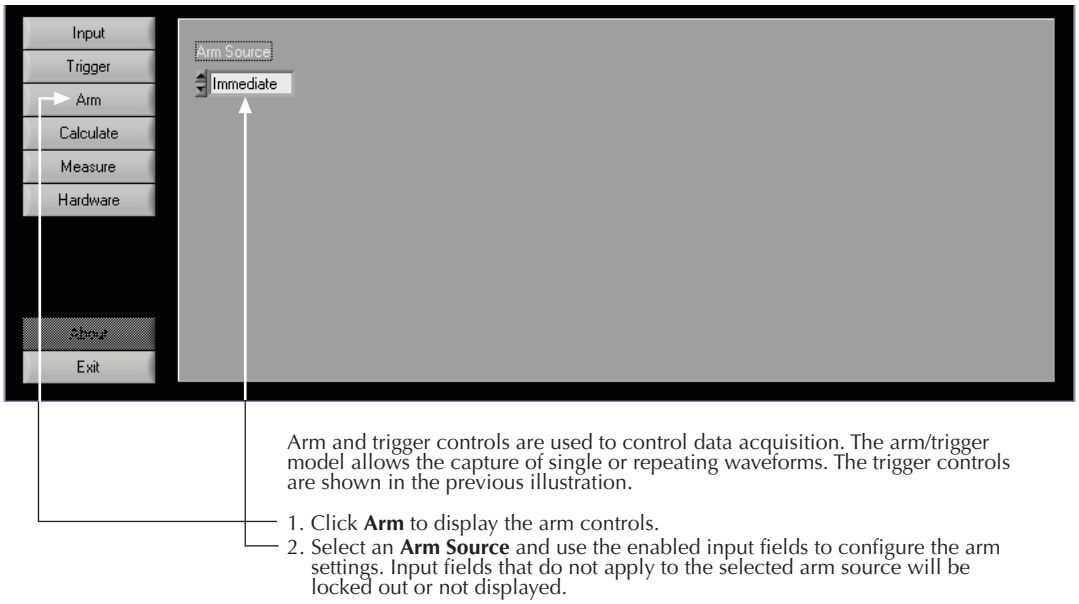
Figure 14-2
KScope: Configuring the Trigger settings



Configure Arm settings

Figure 14-3 explains how to configure the **Arm** settings for the scope.

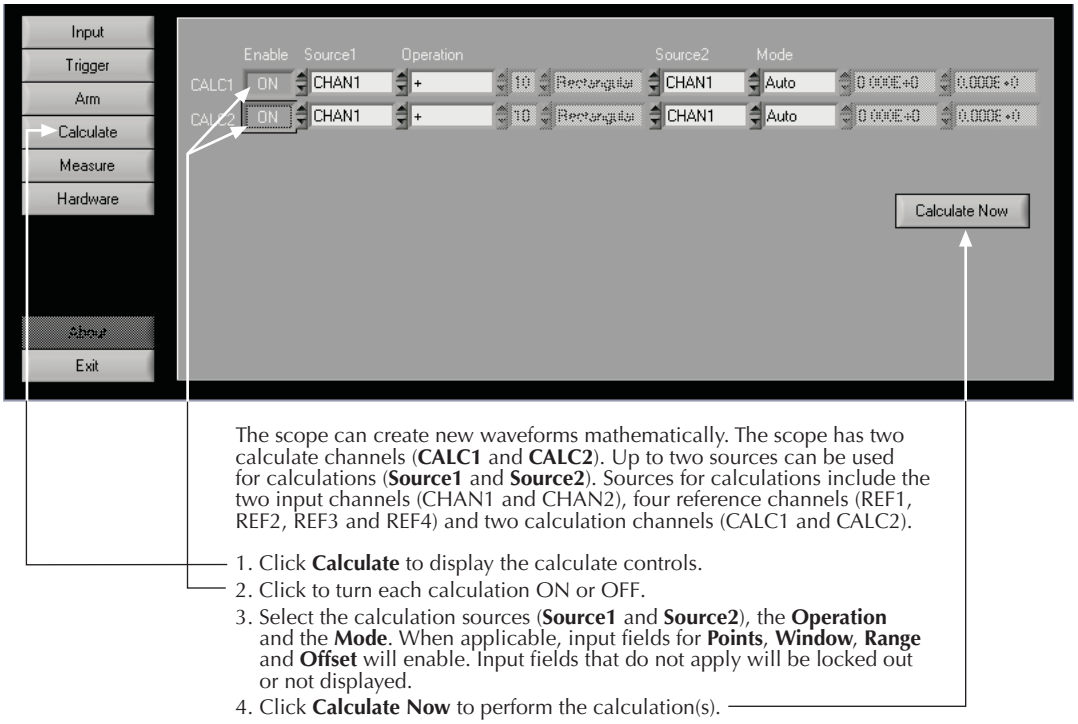
Figure 14-3
KScope: Configuring the Arm settings



Configure Calculate settings

Figure 14-4 explains how to configure the **Calculate** settings for the scope.

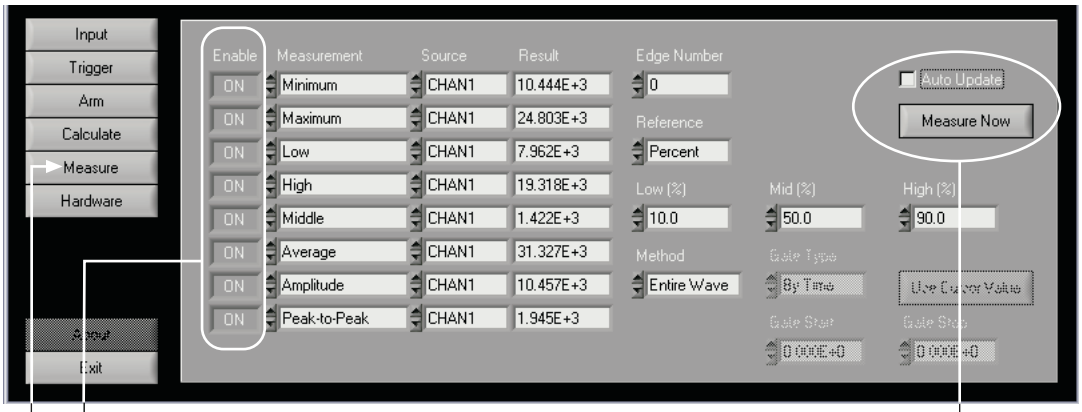
Figure 14-4
KScope: Configuring the Calculate settings



Configure Measure settings

Figure 14-5 explains how to configure the **Measure** settings for the scope.

Figure 14-5
KScope: Configuring the Measure settings



The scope can perform up to eight enabled measurements simultaneously. There are 31 measurement types that can be selected. The source of each measurement can be an input channel (**CHAN1** or **CHAN2**), a reference channel (**REF1**, **REF2**, **REF3** or **REF4**) or a calculation channel (**CALC1** or **CALC2**).

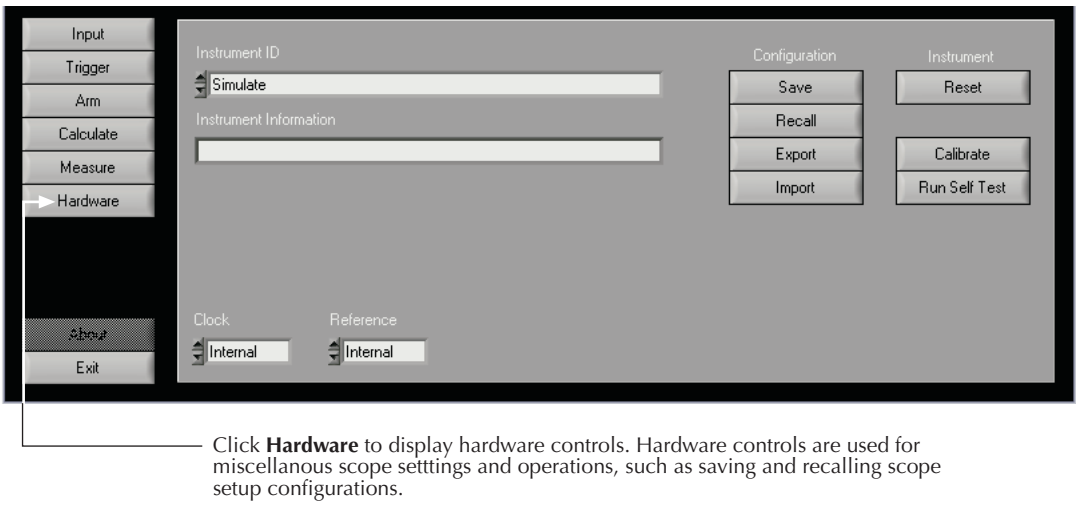
1. Click **Measure** to display the measurement controls.
2. Click to **Enable** (**ON**) or disable (**OFF**) the eight measurements.
3. Select the **Measurement** type and the **Source** for each enabled measurement. Applicable input fields for other measurment settings will be enabled. Input fields and controls that do not apply will be disabled or not displayed.
4. Perform measurements - If **Auto Update** is selected (Ö), measurements will be performed automatically. With **Auto Update** disabled, click **Measure Now** to perform one set of measurements.

   Measured readings are placed in the **Result** fields.

Configure the Hardware settings

Figure 14-6 explains how configure the **Hardware** settings for the scope.

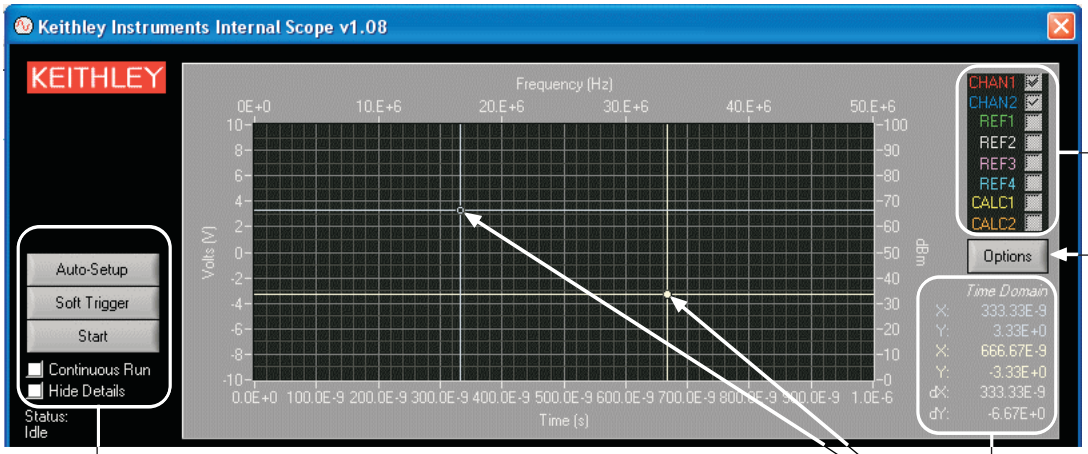
Figure 14-6
KScope: Configuring the Hardware settings



Operate the scope

Figure 14-7 explains how to operate the scope.

Figure 14-7
KScope: Operation



Basic scope operation:

Auto-Setup	Click to automatically adjust scope settings to the input signal(s).
Start	Click to start the scope.
Soft Trigger	Click to trigger a measurement cycle, regardless of the selected trigger source.
Continuous Run	Enable (√) for continuous measurements when the scope is triggered (started).
Hide Details	Enable (√) to hide the scope configuration controls of the GUI.

Scope display:

Select (√) the waveform sources that you wish to display. Waveform sources not enabled in the setup configuration cannot be selected.

Graph options:

Clicking **Options** opens a window to set graph options:

- Set the scales for the X and Y axis of the graph (time domain and frequency domain). Autoscale can be used or they can be set manually.
- Set the two cursor crosspoints to display time or freqency readings. The cursors can also be disabled (not displayed). The readings for the X and Y cursor crosspoint coordinates are displayed below the **Options** button.
- Copy a waveform to a reference channel (REF1, REF2, REF3 or REF4).
- Save a waveform as a .csv file on the computer or your network.

Cursor crosspoint readings (X and Y coordinates):

The readings for the X and Y coordinates for the two cursors are displayed (yellow and blue). A cursor crosspoint is moved by using the mouse to "click and drag" it to the desired location on the graph.

The readings correspond to the selected domain (time or frequency). Also included are the differential readings for the two X coordinates (dX) and the two Y coordinates (dY):

$dX = X_{yellow} - X_{blue}$
$dY = Y_{yellow} - Y_{blue}$

Section 15

Multi-Frequency C-V measurements (Model 4200-CVU)

In this section:

Introduction

The Model 4200-CVU is a multi-frequency (10 kHz to 10 MHz) impedance measurement card that is installed in the Model 4200-SCS mainframe. The AC test signal (10 mV RMS to 100 mV RMS) can be DC voltage biased from -30 V to +30 V.

The CVU measures impedance by sourcing an AC voltage across the device under test (DUT), and then measures the resulting AC current and phase difference. The capacitance and conductance are derived parameters from the measured impedance and phase.

Model 4200-CVU card

Measurement overview

AC impedance measurement (Z_{DUT}) of the device under test (DUT) is performed by sourcing an AC test voltage across the device voltage across the device and measuring the resulting AC current.

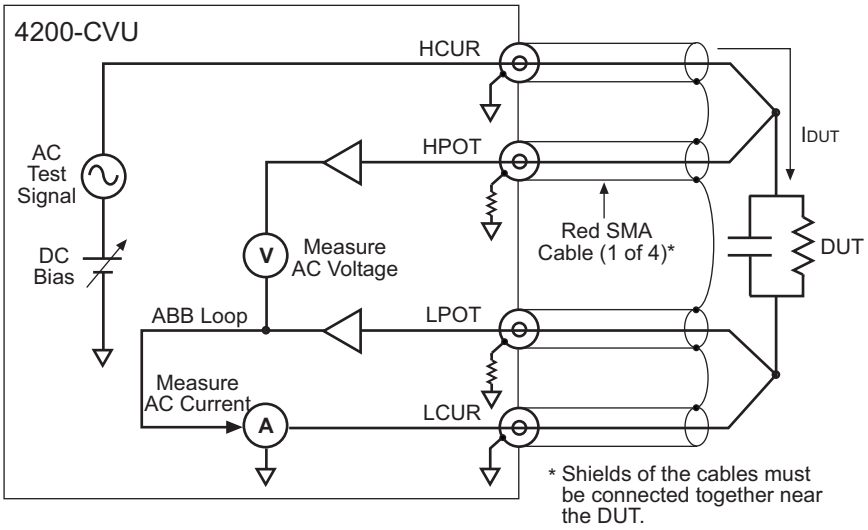
The AC current is measured as shown in Figure 15-1. The Model 4200-CVU uses an auto balance bridge (ABB) technique to achieve accurate impedance measurements. The purpose of the ABB is to create a virtual ground at the DUT to minimize measurement error. Every CVU measurement is taken with ABB active. The ABB will always attempt to lock the low side of the DUT to virtual ground. If the ABB fails to lock:

A. The measurement will still be taken but may be out of specification.
B. The returned data will be flagged and colored yellow in the data sheet.

Most common causes of ABB not locked are as follows:

- Mismatched physical cable lengths
- Mismatched physical cable lengths versus the programmed cable length in Keithley Interactive Test Environment (KITE)
- Improperly torqued SMA cables
- Sub-optimal I-range setting
- Too much parasitic load on the low side of DUT

Figure 15-1
Measurement circuit (simplified)



The capacitive impedance (and conductance) are calculated based on the measured AC impedance and phase.

The capacitance is calculated from the capacitive impedance and the test frequency using the following formula:

$$C_{DUT} = \frac{I_{DUT}}{2\pi f V_{AC}}$$

C_{DUT} = Capacitance of the DUT (f)
f = Test frequency (Hz)
V_{AC} = Measured AC voltage (V)

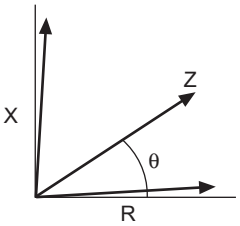
Measurement functions

The Model 4200-CVU can measure the following parameters:

- Z, Theta — Impedance and Phase Angle
- R + jX — Resistance and Reactance
- Cp-Gp — Parallel Capacitance and Conductance
- Cs-Rs — Series Capacitance and Conductance
- Cp-D — Parallel Capacitance and Dissipation Factor
- Cs-D — Series Capacitance and Dissipation Factor

Figure 15-2 shows the vector diagram and fundamental equations for impedance.

Figure 15-2
Vector diagram for impedance (Z)



$$|Z| = \sqrt{R^2 + X^2} \qquad R = Z\cos\theta$$

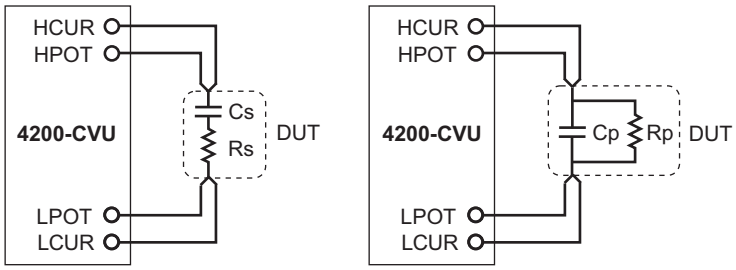
$$Z = R + jX \qquad X = Z\sin\theta$$

$$\theta = arc\tan\left(\frac{X}{R}\right) \qquad Y = \frac{1}{Z} = (G + jB)$$

Z = Impedance
θ = Phase Angle
R = Resistance
X = Reactance
Y = Admittance
G = Conductance

The simplified model of a device-under test (DUT) is a resistor and a capacitor. As shown in Figure 15-3, the Model 4200-CVU can measure the DUT as a series configuration of the resistor-capacitor (RC), or as a parallel RC configuration.

Figure 15-3
Measure models (simplified)



Test signal

The test signal can be set for the following frequencies:

- 10 kHz through 100 kHz in 10 kHz increments
- 100 kHz through 1 MHz in 100 kHz increments

- 1 MHz through 10 MHz in 1 MHz increments

The AC signal output level can be set from 10 mV RMS to 100 mV RMS (1 mV resolution). The output impedance is 100 Ω (typical).

There are three current measurement ranges available to measure current: 1µA, 30µA or 1 mA. With auto range selected, range selection will be performed automatically.

DC bias function and sweep characteristics

The AC test signal can be biased with a static DC level (-30 V to +30 V), or a voltage sweep (up or down). You can also perform a frequency sweep (up or down):

- Figure 15-4 shows an example of DC bias waveform. The DC bias is set to 0 V, but can be set to any valid DC bias level (you specify the number of measurements to perform).
- Figure 15-5 shows an example of DC voltage sweep. You specify the start voltage, stop voltage and step voltage. The number of data (measurement) points is calculated by the Model 4200-CVU.
- Figure 15-6 shows an example of a frequency sweep. You specify the start frequency and the stop frequency; the number of data (measurement) points is calculated by the CVU.

Not shown are the voltage list sweep and the step frequency sweep:

- For the list sweep, you specify the voltage levels for the sweep.
- The step frequency sweep includes voltage stepping. A voltage sweep will be performed for every frequency point.

NOTE *Refer to the User's Manual, Forcing functions and measure options, page 3-14 for details on the bias and sweep forcing functions.*

Figure 15-4
DC Bias waveform (example)

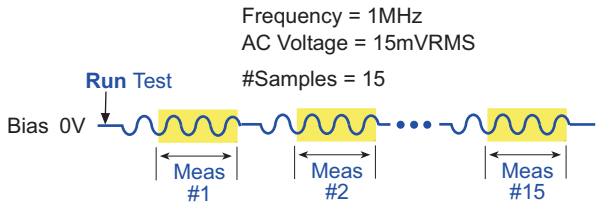


Figure 15-5
DC voltage sweep (example)

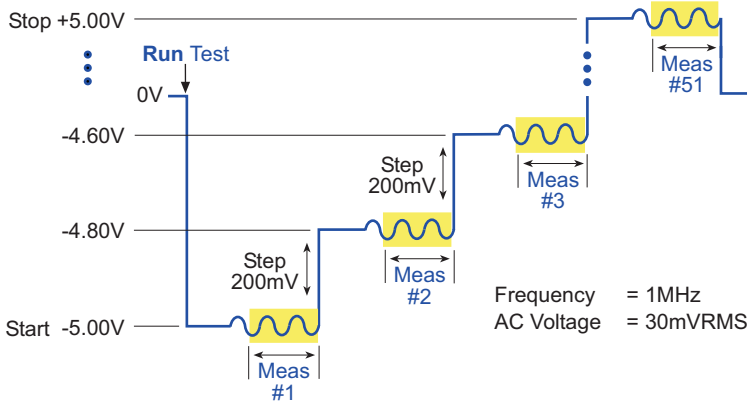
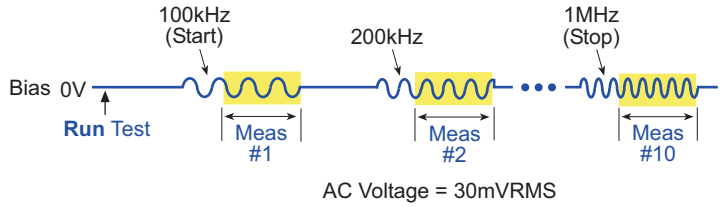


Figure 15-6
Frequency sweep (example)



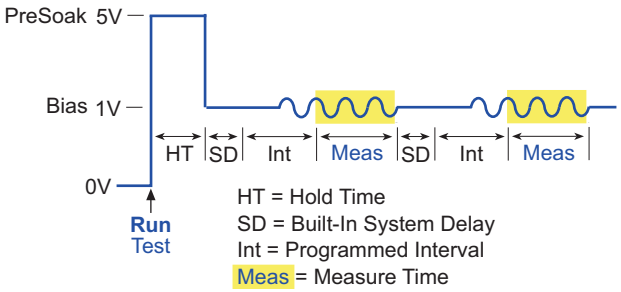
Force-measure timing

Bias function

Timing for the force-measure process for a bias function is shown in Figure 15-7. When the test is started, the following timing sequence takes place.

1. The DC source outputs the PreSoak voltage for the Hold Time period.
2. The DC source goes to the DC Bias voltage.
3. After the built-in system delay and time Interval periods, the Model 4200-CVU performs a measurement. The AC test signal is applied just before the start of the measurement. AC drive is turned off after the measurement is completed.
4. Step 3 is repeated for every measurement.

Figure 15-7
Force-measure timing



Sweep function

Force-measure timing for a sweep function is similar to the timing for a bias function (shown in Figure 15-7), with the following differences:

- The Hold Time is repeated at the beginning each subsequent sweep step.
- A programmed Delay is used in place of the Interval.

Connections

Red SMA cables (male-to-male, 100 Ω) are supplied items with the Model 4200-CVU. These characterized cables must be used for connection to the CVU in order to achieve optimum performance. The CVU is shipped with a four-cable set. The length of each cable is 1.5 m, but 3 m length cables are supplied with the Model 4200-CVU-Prober-Kit.

Connection notes

- Use only the supplied red SMA cables for connections to the Model 4200-CVU. Use either the 1.5 m length cables or the 3 m length cables. Do not use a mix of the two cable lengths.
- The cable length setting in the Definition tab (**Compensation** button) of an ITM must match the length of the SMA cables being used for connections to the CVU. For details, see the User's Manual, KITE ITM configuration, page 3-12.
- After the connection setup is completed, connection compensation must be performed before running any tests. The connection compensation procedure must be performed any time there is a change to the connection setup. For details, see Connection compensation.
- When making connections from the CVU to the device under test (DUT), make sure the shields of the SMA cables are connected together as close as possible to the DUT (see Figure 15-1).

Cables and adapters

Supplied cables and adapters

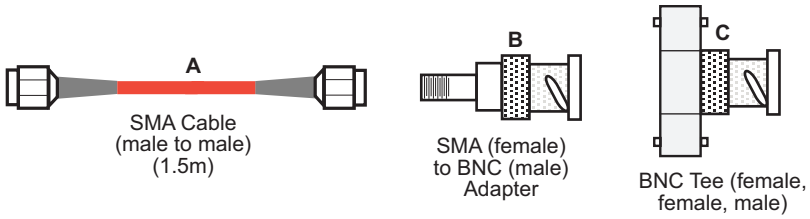
The cables and adapters listed in Table 15-1 are included with every Model 4200-CVU. They are shown in Figure 15-8.

Table 15-1
Supplied cables and adapters

Item*	Qty.	Keithley P/N	Description
A	4	CA-447A	Red SMA cables, male to male, 100Ω, 1.5 m in length
B	4	CS-1247	SMA female to BNC male adapter
C	2	CS-701A	BNC Tee adapter (female, female, male)

* The cable and adapters are shown in Figure 15-8.

Figure 15-8
Supplied cable and adapters



Supplied cables and adapters for the Model 4200-CVU-Prober Kit

The Model 4200-CVU-Prober-Kit was designed to cover a wide variety of prober and manipulator types. For example, some probers have triax feedthroughs on the rear panel, while others have direct BNC connections to the manipulators themselves.

This kit contains a combination of triax and BNC adaptors and barrels that will accommodate most prober connection requirements.

Table 15-2 lists the cables and adapters provided with the prober kit.

Table 15-2
Supplied cables and adapters for the Model 4200-CVU-Prober-Kit

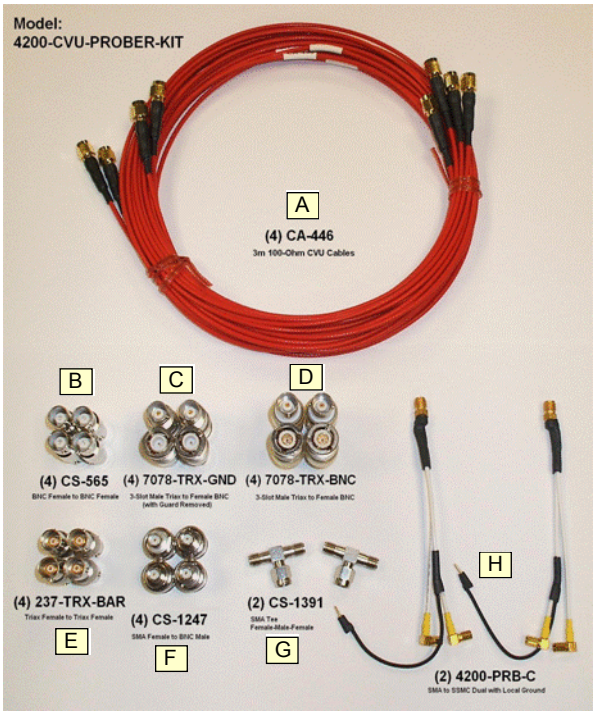
Item*	Qty.	Keithley P/N or Model	Description
A	4	CA-446	Red SMA cables, male to male, 100Ω, 3 m in length
B	4	CS-565	BNC female to BNC female adapter
C	4	7078-TRX-GND	Triax male to BNC female adapter (with guard removed)
D	4	7078-TRX-BNC	Triax male to BNC female adapter
E	4	237-TRX-BAR	Triax female to triax female adapter
F	4	CS-1247	SMA female to BNC male adapter
G	2	CS-1391	SMA Tee adapter (female, male, female)
H	2	4200-PRB-C	SMA to SSMC Dual (with local ground)

* The cables and adapters are shown in Figure 15-9.

The prober kit includes two types of BNC to triax adapters. The Model 7078-TRX-BNC (item D in Table 15-2) has the guard connected to the inner shield of the adapter. The Model 7078- TRX-GND (item C in Table 15-2) has the guard disconnected. Typically, the Model 7078-TRX-BNC is the preferred adaptor in most applications.

When using the Model 4200-PRB-C cables (item H), be sure you jumper the shields together at the probe tips. Each stackable black banana plug is connected to the outer shield of the cables.

Figure 15-9
Model 4200-CVU-Prober-Kit



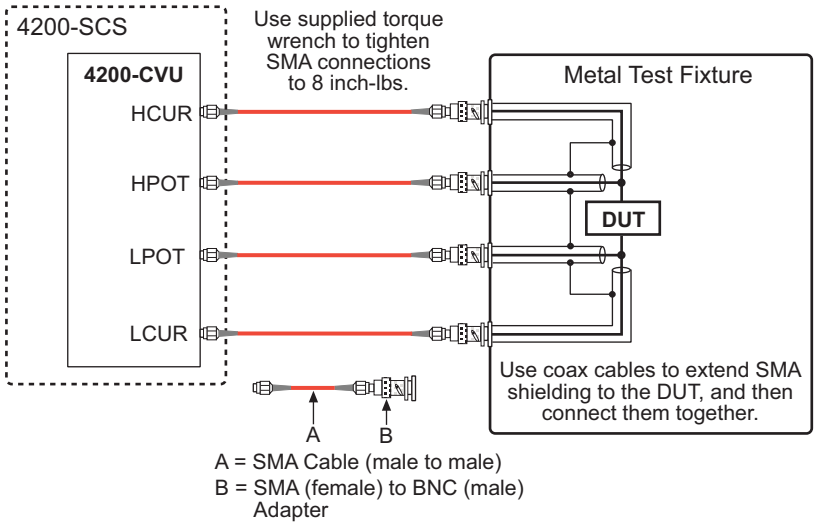
Test connections

Typical test connections to a DUT

NOTE *The shields of the SMA cables must be connected together and extended as far as possible to the device under test (DUT) (see Figure 15-1).*

Figure 15-10 shows typical connections to a DUT installed in a test fixture equipped with female BNC bulkhead connectors. The cables and adapters are supplied with the Model 4200-CVU. Use a conductive test fixture with the bulkhead connectors mounted directly to the test fixture. Do not use insulators between the connectors and test fixture.

Figure 15-10
Test connections to DUT



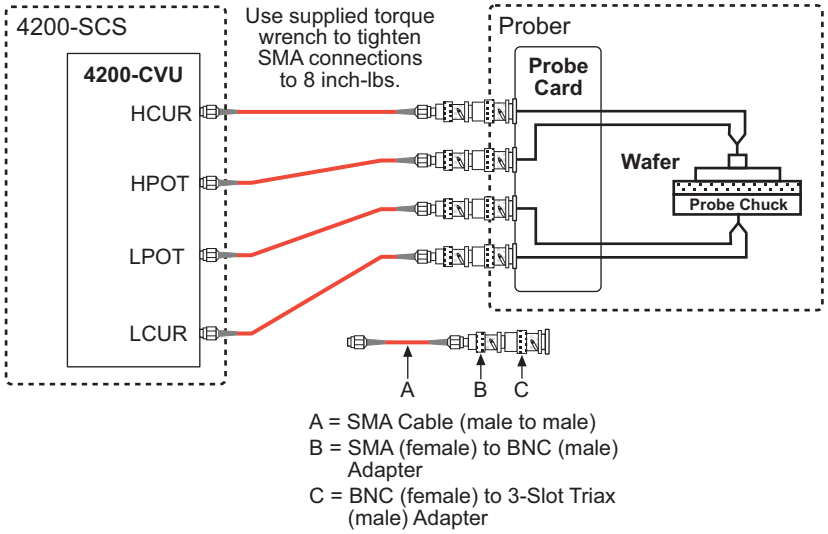
Test connections for a probe card

NOTE *The shields of the SMA cables must be connected together and extended as far as possible to the DUT (see Figure 15-1).*

The Model 4200-CVU-Prober-Kit includes 3-meter SMA cables and an assortment of connection accessories to connect the Model 4200-CVU to a probe card. The supplied items for the kit are listed and described in Table 15-2. The cables and adapters supplied with the CVU (see Table 15-1) and the prober kit will accommodate most connection requirements.

Figure 15-11 shows typical test connections for a probe card that use triax (female) connectors. The prober kit includes two types of BNC to triax adapters. The Model 7078-TRX-BNC (item D in Table 15-2) has the guard connected to the inner shield of the adapter. The Model 7078-TRX-GND (item C in Table 15-2) has the guard disconnected. Typically, the Model 7078-TRX-BNC is the preferred adaptor in most applications.

Figure 15-11
Typical connections to a probe card



NOTE *If the probe card uses BNC (female) connectors, the BNC to triax adapters (C) are not necessary.*

Test connections for a switch matrix

NOTE *The shields of the SMA cables must be connected together and extended as far as possible to the DUT (see Figure 15-1).*

Figure 15-12 and Figure 15-13 shows typical connection schemes for a switch system using a Model 707A switching mainframe with the Model 7174A Matrix Card installed. Figure 15-12 shows connections for local (2-wire) sensing, and Figure 15-13 shows connections for remote (4-wire) sensing.

Figure 15-12
Test connections for a switch matrix - local (2-wire) sensing

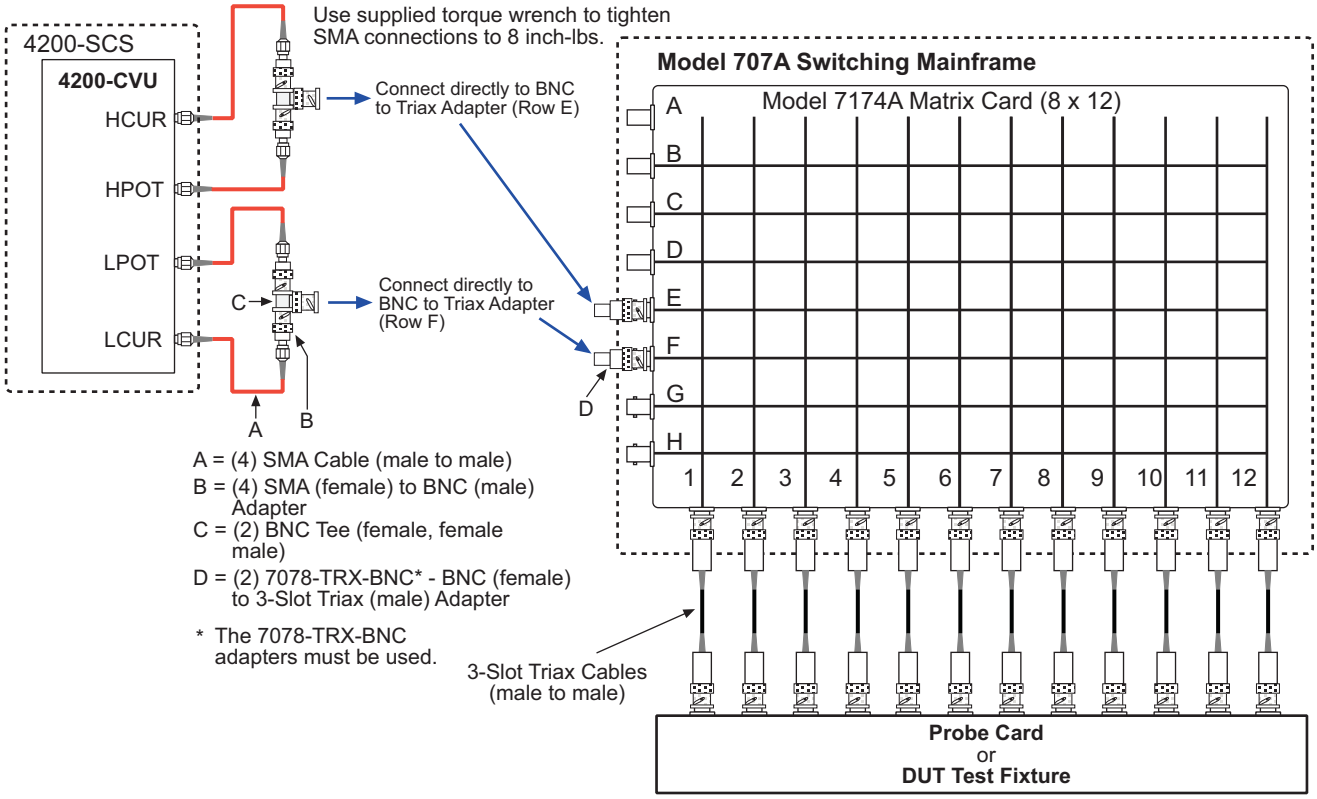
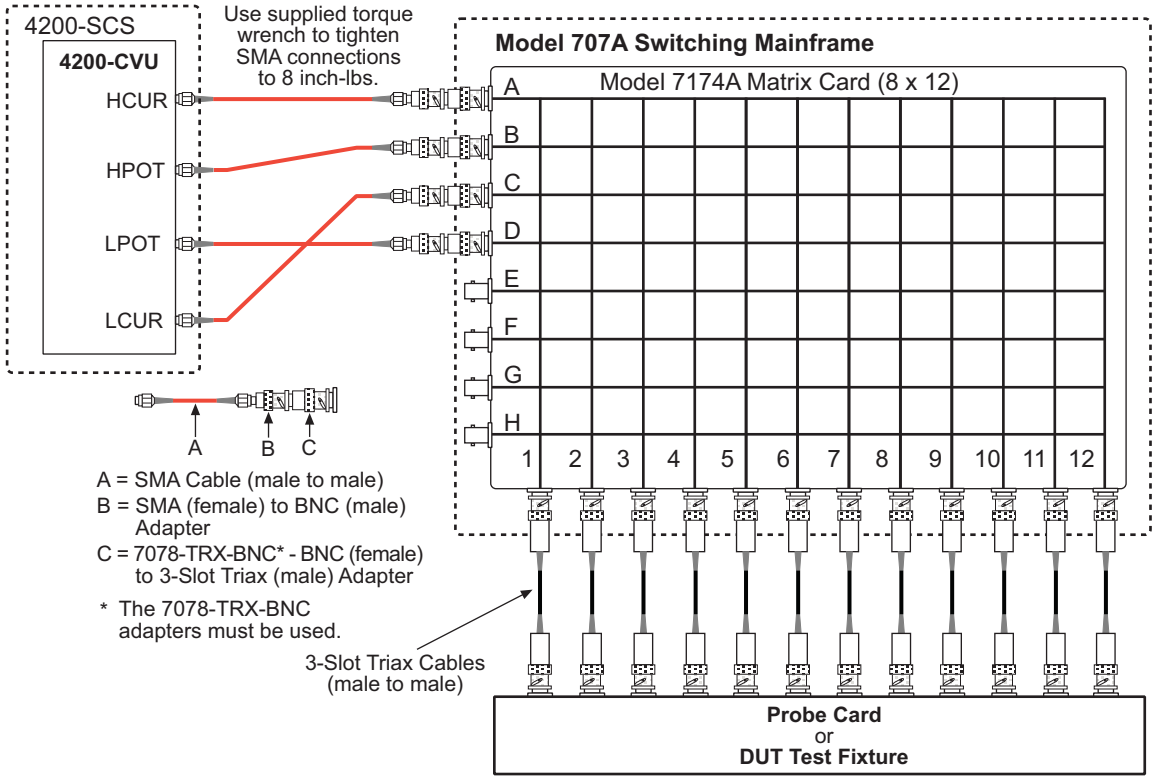


Figure 15-13
Test connections for a switch matrix - remote (4-wire) sensing



NOTE *The Model 7072 matrix card can also be used for C-V testing. However, you must use Rows G and H, and local (2-wire) sensing.*

Figure 15-12 shows connections for local (2-wire) sensing. It shows the Model 4200-CVU connected to Rows E and F of the matrix. This is the connection scheme for the CVU_ivcvswitch project plan. For details, see CVU_ivcvswitch.

The SMA cables and adapters shown in the two drawings are supplied with either the CVU or the Model 4200-CVU-Prober-Kit. Not supplied are the Triax and BNC cables.

The prober kit includes two types of BNC to triax adapters that connect directly to the rows of the matrix. The Model 7078-TRX-BNC (item D in Table 15-2) has the guard connected to the inner shield of the adapter. The Model 7078-TRX-GND (item C in Table 15-2) has the guard disconnected.

NOTE *The Model 7078-TRX-BNC adapters must be used in order to extend SMA shielding through the matrix card.*

KCON configuration

The connections to the matrix card must match the Keithley CONfiguration utility (KCON) settings for the switching matrix:

- The KCON Properties tab for the switching matrix is used to add and configure the switch matrix (for example, Model 707A Switching Mainframe) for the test system.
- The KCON Properties tab for the matrix card is used to add and configure the matrix card (Model 7174A Matrix Card) for the test system.

GPIB connection

The switching matrix is an external instrument that is controlled through the General Purpose Interface Bus (GPIB). Make sure to connect a GPIB cable from the switching mainframe to the Model 4200-SCS (see GPIB connection).

Confidence Check

Confidence Check is a diagnostic tool that allows you to check the integrity of open and short connections and connections to a device-under test (DUT). When the Model 4200-CVU is connected to the DUT, the Confidence Check displays the measured readings in real time. This allows Confidence Check to be used as a C-V meter to perform quick and accurate measurements without using an ITM or UTM. The measurement settings (AC drive, DC drive, and speed) can be set from the graphical user interface (GUI) for Confidence Check.

An open or short confidence check performs a measurement on the high and the low sides of the test circuit. The simplest to use is the Open Check since it is the easiest to set up. An Open Check will catch most of the problems. The Short Check can be used if Open Check doesn't detect a problem.

The GUI for confidence check is opened from the Tools menu on the KITE menu bar (see Figure 15-14). As shown in the GUI window, real-time measurements will be performed and displayed. It can, for example, be used to confirm that contact has been made with the pads on a wafer. The measurements are independent of the open and short confidence checks.

NOTE *Confidence Check can be used with a switching matrix. Connect the switching matrix to the CVU and DUT (or short) as explained in Test connections for a switch*

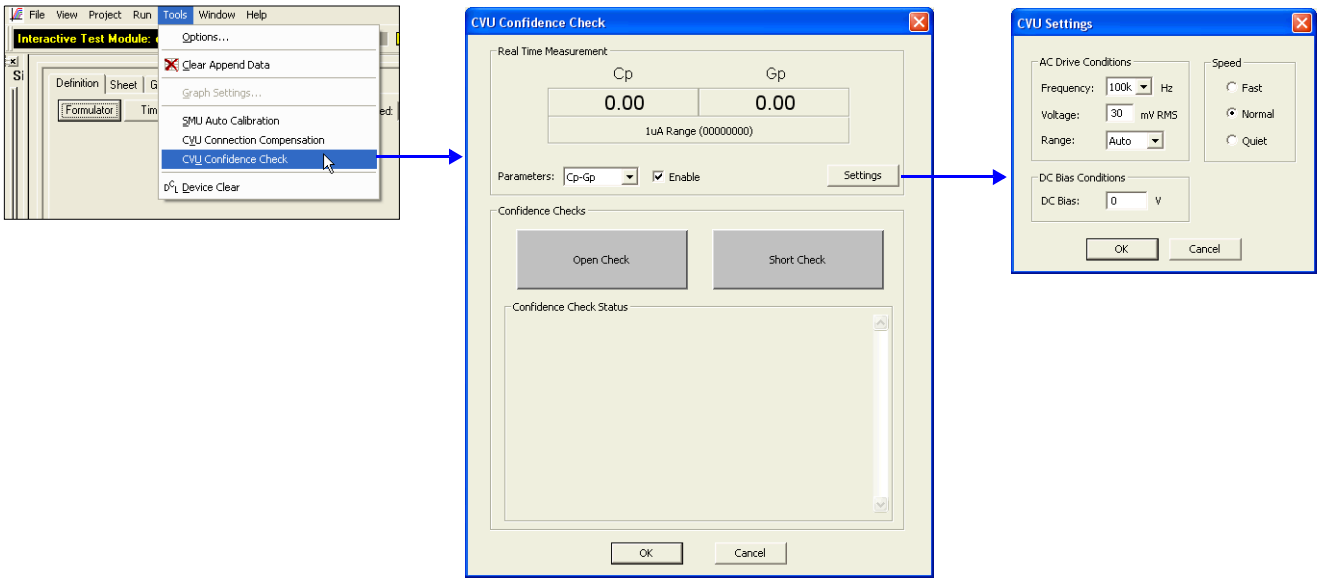
matrix. When performing real-time measurements (or Short Check), close the matrix switches to connect the CVU to the DUT (or short). When performing the Open Check, also open the matrix switches, but lift the probes or disconnect the DUT.

The switches can be controlled from the switching mainframe or a User Test Module (UTM) can be used to open/close the switches. The connect-cv UTM in the CVU_ivcvswitch project can be used to control switching (see connect and connect-cv UTMs).

The GUI setup for Confidence Check includes the following:

- Select the measurement type from the drop-down menu for **Parameters**.
- **Enable** (check) the measurement type.
- Click the **Settings** button to open the CVU Settings window. Set the AC Drive Conditions, DC Bias Conditions, and Speed.

Figure 15-14
Confidence check



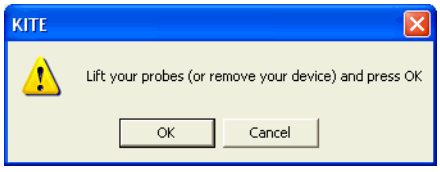
Open Check and Short Check

As shown in the GUI for CVU Confidence Check, there are two checks: Open Check and Short Check. With the Model 4200-CVU connected to the DUT (wafer), perform the following steps to perform a confidence check:

1. In the GUI window, click the **Open Check** or **Short Check** button to display one of the prompts shown in Figure 15-15.
2. Perform the action indicated in the prompt and then click **OK** to perform the check.
3. If the check passes, the check button turns green (as shown in Figure 15-16). If the results are not very good, the button will turn yellow. If the check fails, the check button turns red (as shown in Figure 15-17). An analysis of the failure is provided in the GUI window. Use the scroll bar to display troubleshooting information.

Figure 15-15
Prompts for confidence checks

Open Check:



Short Check:

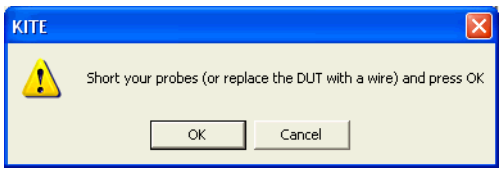
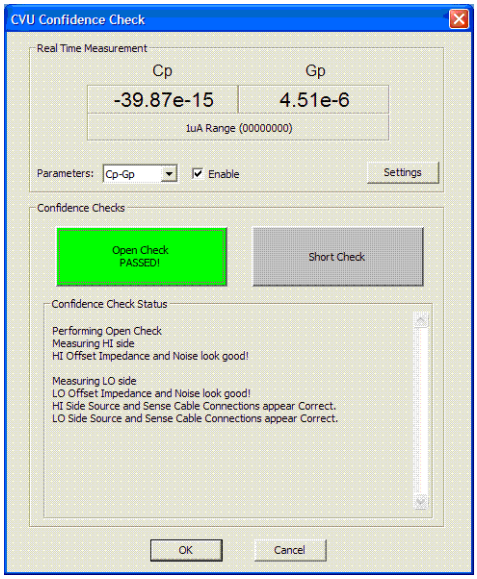


Figure 15-16
Pass status



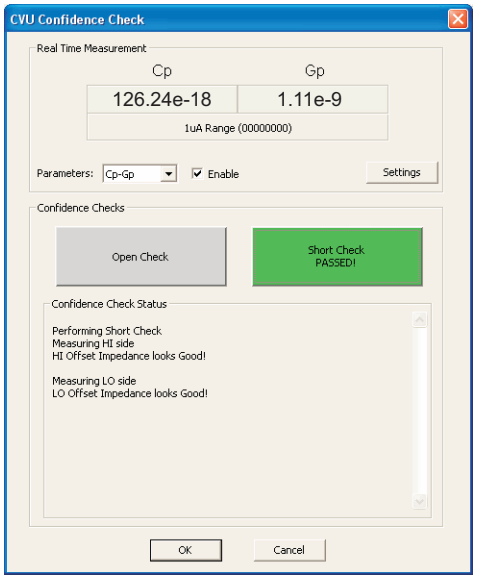
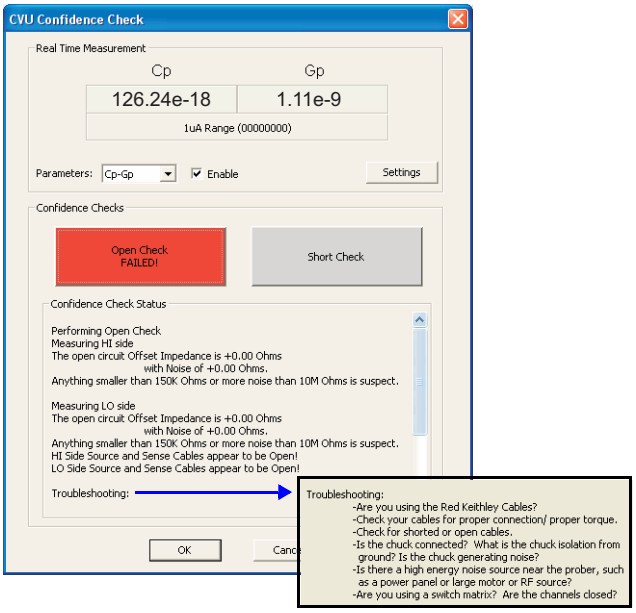
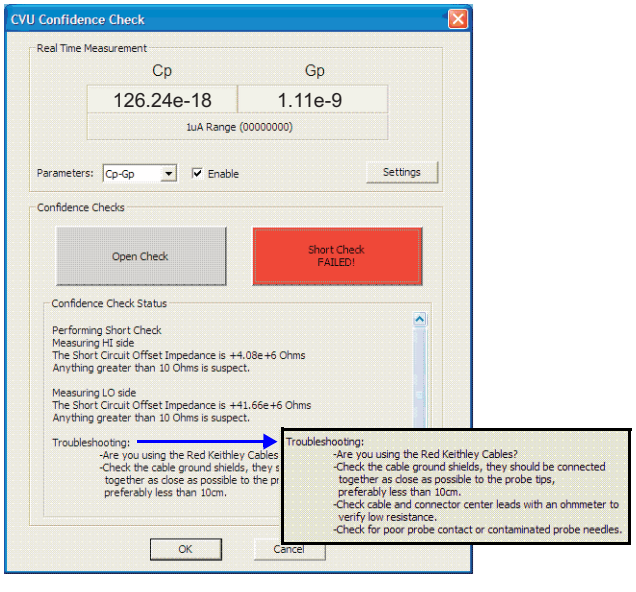


Figure 15-17
Fail status





Connection compensation

Offset and gain errors caused by the connections between the Model 4200-CVU and the device under test (DUT) can be corrected by using Connection Compensation. Correction is a two-part process:

1. Connection compensation data is generated for open, short, and load conditions. The compensation values are stored in tables.
2. The correction values for open, load, and short must be enabled before running the ITM.

When an ITM is run, each measurement will factor in the enabled compensation values. If open, short and load compensation are disabled, the compensation values will not be used by the ITM.

NOTE *Connection compensation should be performed anytime the connection setup is changed or disturbed. Changes in temperature or humidity do not affect connection compensation.*

Guidelines to determine required correction

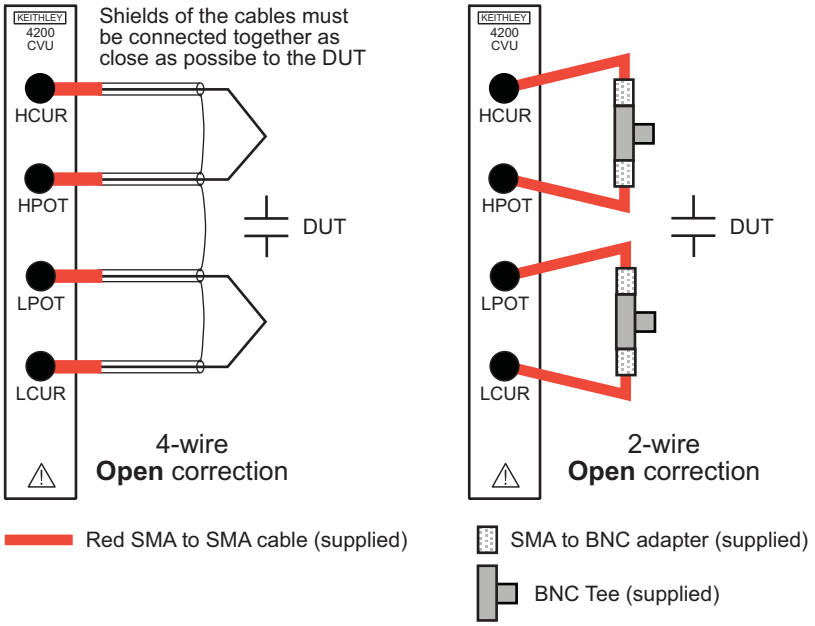
It is good practice to perform compensation for all three connection conditions (open, short, and load), but may not be necessary. Use the following general guidelines to determine which correction needs to be performed:

- **Open** correction Offset correction for small capacitance (>1 M Ω, large impedance)
- **Short** correction Offset correction for large capacitance (<10 Ω, small impedance)
- **Load** correction Resistive load correction for gain error

Connections for connection compensation

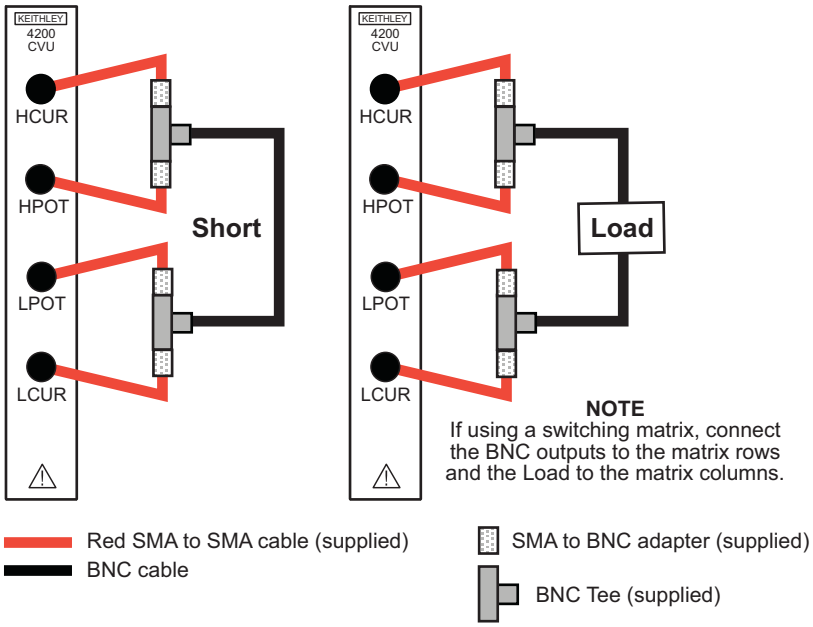
The connections for Open is shown in Figure 15-18. For remote (4-wire) sensing, HCUR and HPOT must be connected together, and LCUR and LPOT must be connected together. Also, the shields of the four SMA cables must be connected together as close as possible to the DUT.

Figure 15-18
Connections for Open connection compensation



The connections for Short and Load are shown in Figure 15-19. For Load correction, Figure 15-12 provides details on connections for the switching matrix.

Figure 15-19
Connections for Short and Load connection compensation



Generating connection compensation data

The graphical user interface (GUI) to generate connection compensation data is opened from the **Tools** menu, as shown in Figure 15-20. The GUI is shown in Figure 15-21.

Notice in the GUI that the SMA cable length being used must be set:

- **0 M** Use when performing measurements at the terminals of the Model 4200-CVU (no cabling). This effectively disables compensation.
- **1.5 M** Standard red SMA cables (1.5 m) supplied with the CVU.
- **3.0 M** Red SMA cables (3.0 m) supplied with the prober kit. This setting can also be used when using a switching matrix.

The above setting will become the default length setting in the ITMs.

NOTE *If using a switching matrix, close the matrix switches that will connect the CVU to the Open, Short or Load. The switches can be closed from the switching mainframe or a UTM can be used to close the switches. The connect-cv UTM in the CVU_ivcvswitch project can be used to control switching (see connect and connect-cv UTMs). The ConnectPins user library of the Matrixulib User Library can also be used.*

Figure 15-20
Opening the GUI

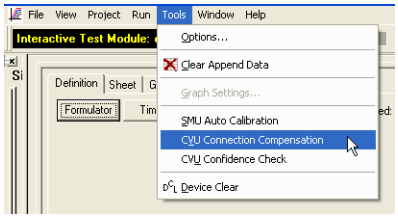
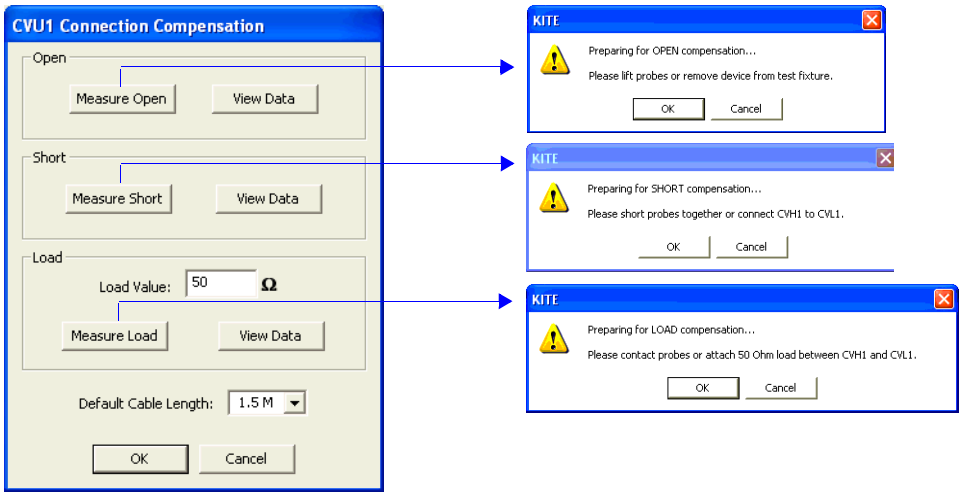


Figure 15-21
GUI for generating connection compensation data



NOTE *Refer to Figure 15-21 for the following procedures:*

Open correction

Perform the following steps to generate open correction values:

1. Set up the open configuration as shown in Figure 15-18.
2. Set the SMA cable length.
3. Click the **Measure Open** button to display the connection prompt.

4. When ready, click **OK** to perform the correction.

Short correction

Perform the following steps to generate short correction values:

1. Set up the short configuration as shown in Figure 15-19.
2. Set the SMA cable length.
3. Click the **Measure Short** button to display the connection prompt.
4. When ready, click **OK** to perform the correction.

Load correction

Perform the following steps to generate load correction values:

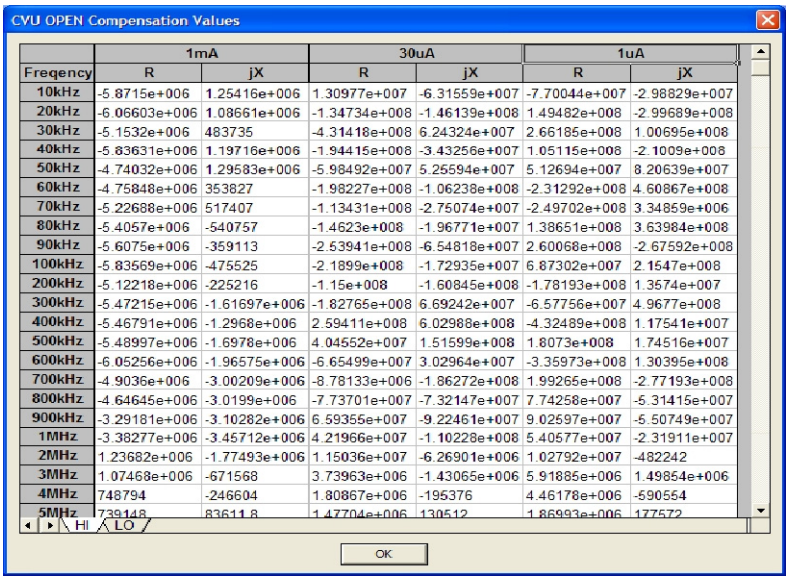
1. Set up the load configuration as shown in Figure 15-19.
2. Set the standard **Load Value** (in ohms) that is being used.
3. Set the SMA cable length.
4. Click the **Measure Load** button to display the connection prompt.
5. When ready, click **OK** to perform the correction.

Compensation data

The compensation values for open, short, and load are stored in tables. Whenever compensating values are generated, they will overwrite the old data in the table(s).

The values are displayed by clicking a **View Data** button in the GUI (which is shown in Figure 15-21). As shown in Figure 15-22, R and jX open compensation values are listed for every test frequency and measurement range. Notice that there is a tab (HI) for the high side values, and a tab (LO) for the low side values. There is a similar table for short and load.

Figure 15-22
Open compensation values (example)



	1mA		30uA		1uA	
Freqency	R	jX	R	jX	R	jX
10kHz	-5.8715e+006	1.25416e+006	1.30977e+007	-6.31559e+007	-7.70044e+007	-2.98829e+007
20kHz	-6.06603e+006	1.08661e+006	-1.34734e+008	-1.46139e+008	1.49482e+008	-2.99689e+008
30kHz	-5.1532e+006	483735	-4.31418e+008	6.24324e+007	2.66185e+008	1.00695e+008
40kHz	-5.83631e+006	1.19716e+006	-1.94415e+008	-3.43256e+007	1.05115e+008	-2.1009e+008
50kHz	-4.74032e+006	1.29583e+006	-5.98492e+007	5.25594e+007	5.12694e+007	8.20639e+007
60kHz	-4.75848e+006	353827	-1.98227e+008	-1.06238e+008	-2.31292e+008	4.60867e+008
70kHz	-5.22688e+006	517407	-1.13431e+008	-2.75074e+007	-2.49702e+008	3.34859e+006
80kHz	-5.4057e+006	-540757	-1.4623e+008	-1.96771e+007	1.38651e+008	3.63984e+008
90kHz	-5.6075e+006	-359113	-2.53941e+008	-6.54818e+007	2.60068e+008	-2.67592e+008
100kHz	-5.83569e+006	-475525	-2.1899e+008	-1.72935e+007	6.87302e+007	2.1547e+008
200kHz	-5.12218e+006	-225216	-1.15e+008	-1.60845e+008	-1.78193e+008	1.3574e+007
300kHz	-5.47215e+006	-1.61697e+006	-1.82765e+008	6.69242e+007	-6.57756e+007	4.9677e+008
400kHz	-5.46791e+006	-1.2968e+006	2.59411e+008	6.02988e+008	-4.32489e+008	1.17541e+007
500kHz	-5.48997e+006	-1.6978e+006	4.04552e+007	1.51599e+008	1.8073e+008	1.74516e+007
600kHz	-6.05256e+006	-1.96575e+006	-6.65499e+007	3.02964e+007	-3.35973e+008	1.30395e+008
700kHz	-4.9036e+006	-3.00209e+006	-8.78133e+006	-1.86272e+008	1.99265e+008	-2.77193e+008
800kHz	-4.64645e+006	-3.0199e+006	-7.73701e+007	-7.32147e+007	7.74258e+007	-5.31415e+007
900kHz	-3.29181e+006	-3.10282e+006	6.59355e+007	-9.22461e+007	9.02597e+007	-5.50749e+007
1MHz	-3.38277e+006	-3.45712e+006	4.21966e+007	-1.10228e+008	5.40577e+007	-2.31911e+007
2MHz	1.23682e+006	-1.77493e+006	1.15036e+007	-6.26901e+006	1.02792e+007	-482242
3MHz	1.07468e+006	-671568	3.73963e+006	-1.43065e+006	5.91885e+006	1.49854e+006
4MHz	748794	-246604	1.80867e+006	-195376	4.46178e+006	-590554
5MHz	739148	83611.8	1.47704e+006	130512	1.86993e+006	177572

HI LO

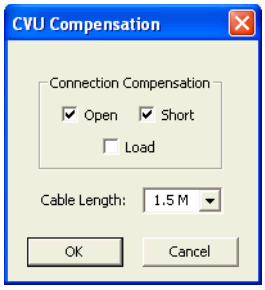
Enabling compensation

The above procedures to generate compensation data do not perform compensation. With open, short, and load correction enabled, compensation will be performed during the execution of an ITM.

Correction is enabled from the CVU Compensation window shown in Figure 15-23. This example shows that correction is enabled for Open and Short. This window is opened by clicking the **Compensation** button in the Definition tab for an ITM. Figure 15-37 shows the location of this button.

Make sure the **Cable Length** setting in the compensation window is the same as the cable setting in the GUI for generating connection compensation data (see Figure 15-21).

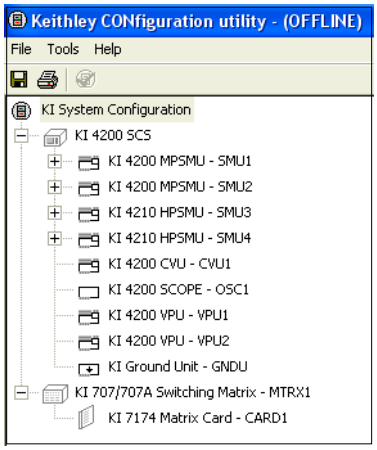
Figure 15-23
Enabling compensation



KCON system configuration

The Keithley Configuration (KCON) is opened by double-clicking the **KCON** icon on the desktop. Figure 15-24 shows an example configuration that includes Model 4200-SMUs, a Model 4200-CVU, and a Model 707A switching mainframe (with Model 7174A matrix card installed). Two SMUs, the CVU, and a switching matrix are required to support the following project plan: CVU_ivcvswitch.

Figure 15-24
KCON system configuration (typical)



KCON configuration for C-V testing using the Model 4200-CVU includes the following:

- Use the Properties & Connections tab to view the instrument properties and the present connection settings for the switching matrix.
- Use the Properties tab for the switching matrix to set the GPIB Address, Instrument Connection Scheme (connections and sensing) and Switch Cards.
- Use the Properties tab for the matrix card to set the terminals for the rows or columns of the matrix.

Properties & Connections tab

Figure 15-25 shows the Properties & Connections tab for the Model 4200-CVU. This tab is displayed by clicking **KI 4200 CVU - CVU1** in the KI System Configuration navigator located on the left side of the KCON window. As shown, this tab consists of three areas: Instrument Properties, Matrix Connections, and Self Test:

Instrument Properties

Provides general information about the Model 4200-CVU, including the slot location of the card, firmware and hardware versions, serial number, and calibration dates.

Matrix Connections

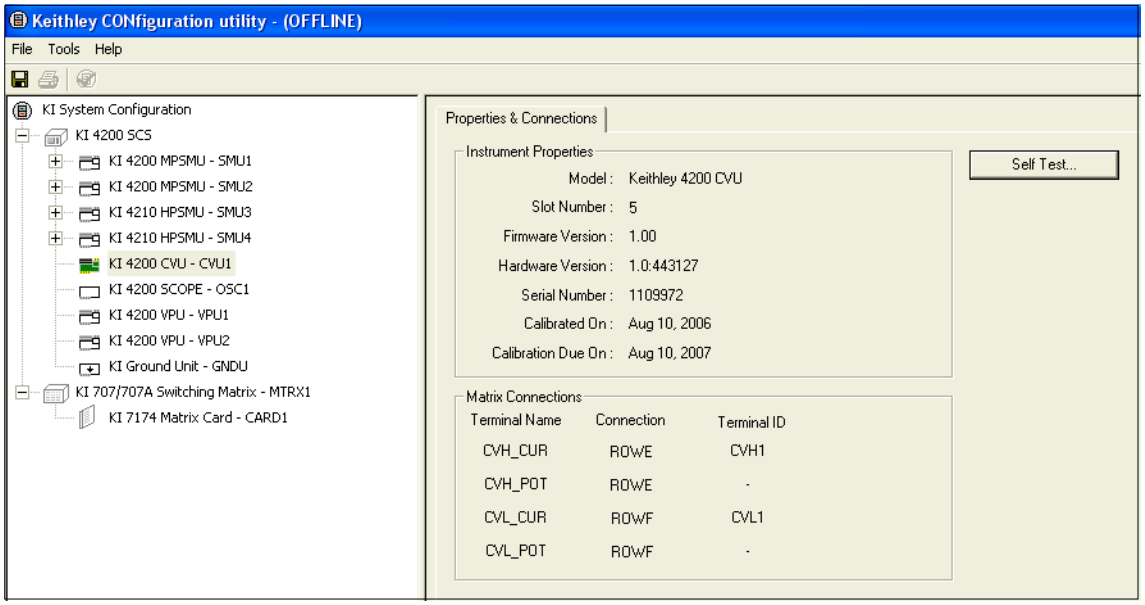
Shows the present Model 4200-CVU connection settings for the switching matrix. The connection settings in Figure 15-25 indicate that the Model 4200-CVU is connected to rows E and F of the matrix. These switch matrix settings are made from the Properties tab for the matrix card.

Self Test

The Self Test button starts the Self Test utility, which aids in troubleshooting potential hardware problems. When you run Self Test, the Model 4200-CVU performs several internal performance

checks to determine if it is operating correctly. No external equipment is required. When the Self Test process finishes, a pass-fail window appears.

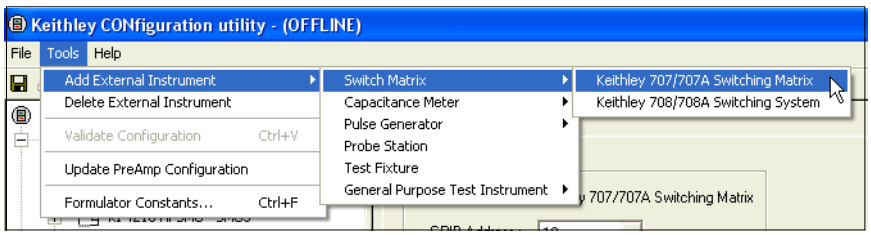
Figure 15-25
KCON: Properties & Connections tab



Properties tab for the switching matrix

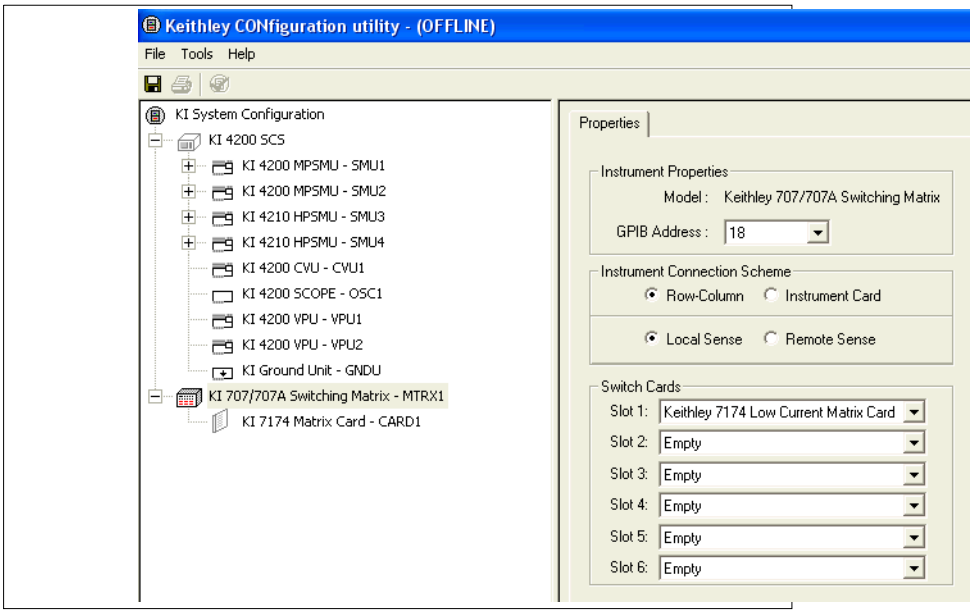
The configuration navigator in Figure 15-25 shows that the system includes a Model 707/707A Switching Matrix that has a Model 7174 Matrix Card installed. Figure 15-26 shows how an external switching matrix is added to the system using the **Tools** menu.

Figure 15-26
KCON: Adding an external switch matrix to the system



The Properties tab for the Model 707/707A Switching Matrix is shown in Figure 15-27. This tab is displayed by clicking **KI 707/707A Switching Matrix - MTRX1** in the system configuration navigator located on the left side of the KCON window. As shown, this tab consists of three areas: Instrument Properties, Instrument Connection Scheme, and Switch Cards.

Figure 15-27
KCON: Properties window for the Model 707/707A Switching Matrix



Instrument Properties

The external switching matrix is a GPIB instrument and needs to be set to a **GPIB Address** (0 to 30) that is not being used by any other GPIB instrument in the system. Note that the GPIB is only used for external instruments.

Instrument Connection Scheme

This area of the tab includes settings for Model 4200-CVU connections to the matrix card, and sense selection:

- Row-Column or Instrument Card:
  - The **Row-Column** setting is used when the Model 4200-CVU terminals are connected to the rows of the matrix card. The DUT is connected to the columns of the matrix card.
  - The **Instrument Card** setting is used when both the Model 4200-CVU and DUT are connected to the columns of the matrix card. Matrix rows are not used for connections.
- Local Sense or Remote Sense:
  - The **Local Sense** setting is used for 2-wire connections.
  - The **Remote Sense** setting is used for 4-wire connections.

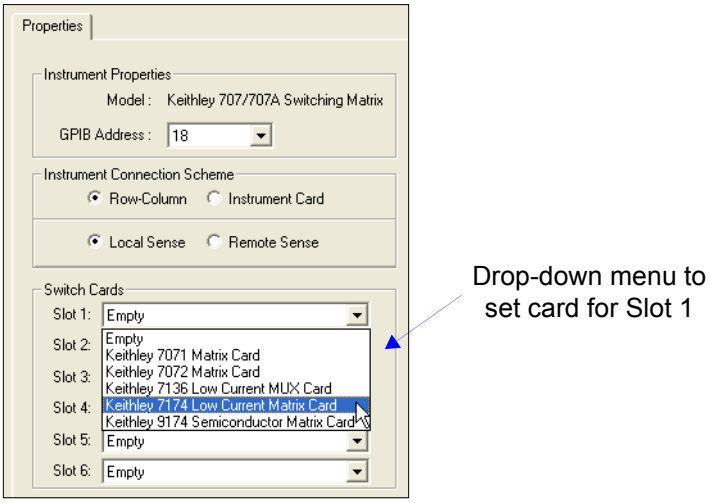
NOTE *The CVU_ivcvswitch project uses row-column connections and local (2-wire) sensing as shown in Figure 15-27. For details, see CVU_ivcvswitch.*

Connections provides details on making Row-Column connections to the Model 4200-CVU for both local (2-wire) and remote (4-wire) sensing.

Switch Cards

This area of the tab shows the switch card settings for the switch matrix. Figure 15-27 shows that a Model 7174 Low Current Matrix Card is selected for slot 1. Figure 15-28 shows the drop-down menu used for switch card settings.

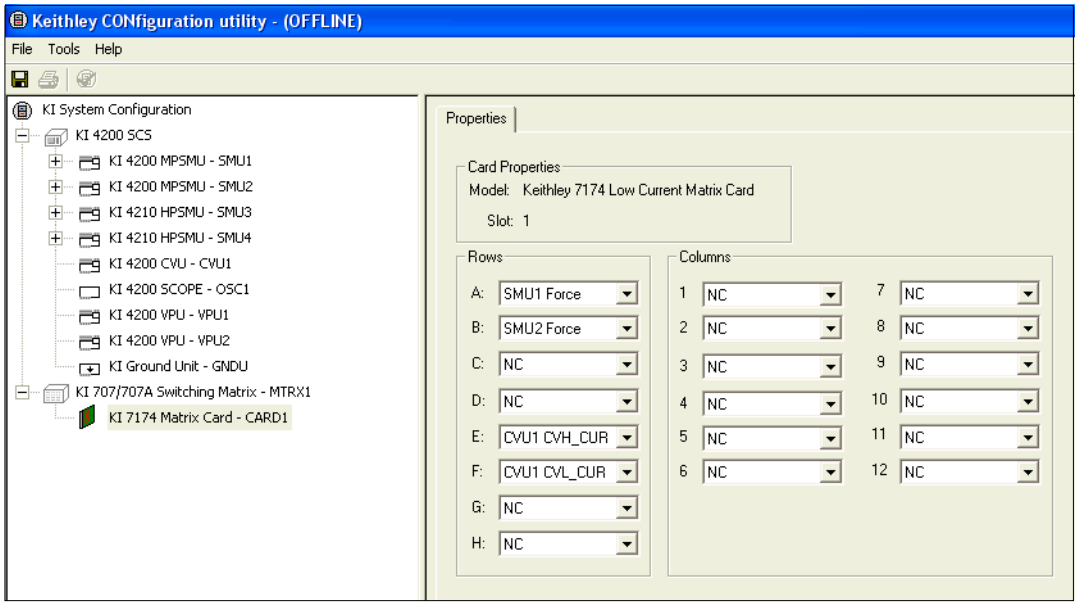
Figure 15-28
KCON: Card menu for Slot 1 of Model 707/707A Switching Matrix



Properties tab for the matrix card

The Properties tab for the Model 7174 Matrix Card is shown in Figure 15-29. This tab is displayed by clicking **KI 7174 Matrix Card - CARD1** in the system configuration navigator located on the left side of the KCON window. This tab is used to set the terminals for the rows and columns of the matrix.

Figure 15-29
KCON: Properties tab for the Model 7174 Matrix Card (local sensing)



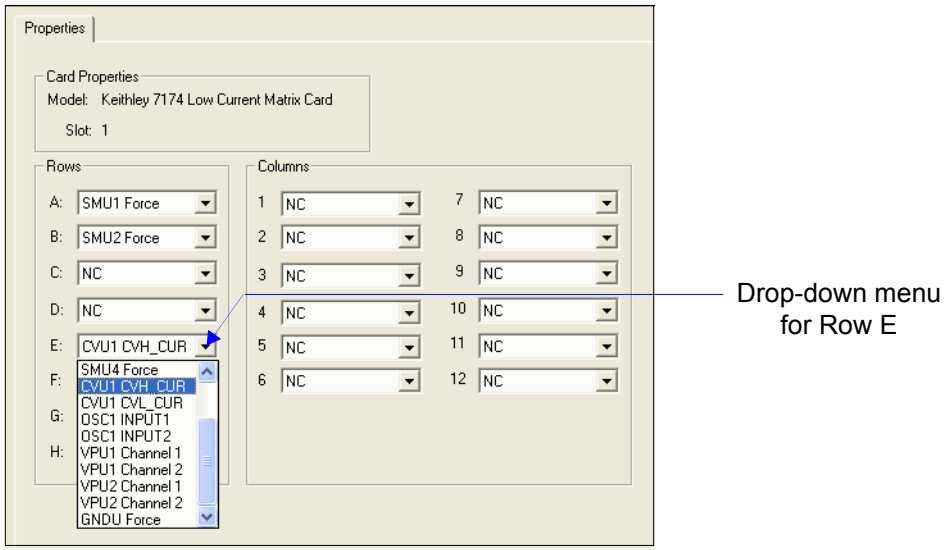
Local (2-wire) sensing

The Properties tab shown in Figure 15-28 is configured for Local Sense and Row-Column terminals. Local sensing is used for the CVU_ivcvswitch project (see CVU_ivcvswitch). These settings are made from the switching matrix Properties tab (see Instrument Connection Scheme).

- **Rows** Figure 15-29 shows that Row E is set for **CVU1 CVH_CUR** and Row F is set for **CVU1 CVL_CUR**. Figure 15-30 shows the drop-down menu used for row/column terminal settings.
- **Columns** For the system configuration shown in Figure 15-24, the only valid setting for the columns is **NC** (no connection). If a test fixture (external instrument) is added to the system, valid settings for the columns will include the test fixture pin numbers.

NOTE *With the Instrument Connection Scheme set for Instrument Card, terminal settings are made for the columns. The rows are fixed at NC (no connection).*

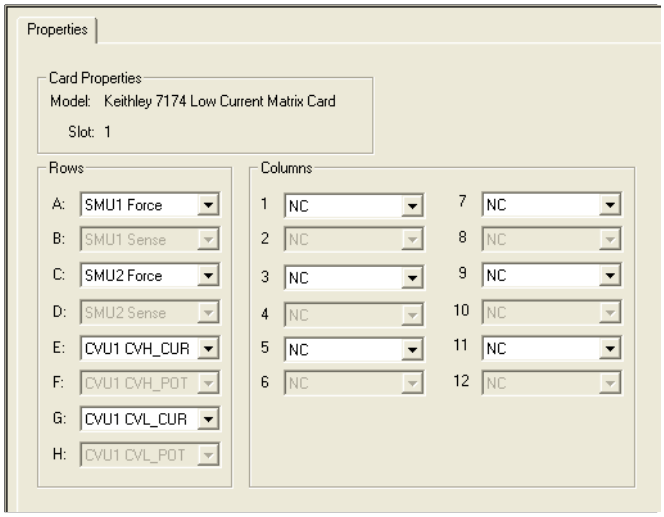
Figure 15-30
KCON: Card menu for Slot 1 of Model 707/707A Switching Matrix



Remote (4-wire) sensing

The Properties tab shown in Figure 15-31 is configured for Remote Sense and Row-Column terminals. As shown, twice as many matrix rows are needed to accommodate all instrument terminals.

Figure 15-31
KCON: Properties tab for the Model 7174 Matrix Card (remote sensing)



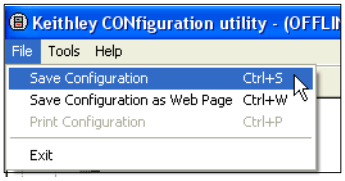
Saving the configuration

Before saving the configuration, validate the KCON configuration using the Tools button on the KCON menu bar:

Click **Tools** > Click **Validate Configuration**

To save the configuration, use the **File** menu shown in Figure 15-32 to save the KCON configuration. If the configuration is not saved, the will be lost when KCON is closed.

Figure 15-32
KCON: Saving the configuration



NOTE *Instructions and examples on how to create CVU test in ITMs are covered in How to perform a C-V test on my device.*

C-V project plans

Project plans overview

A project plan contains the interactive test module (ITMs) for a testing application. The twelve project plans that support Model 4200-CVU testing are summarized in Table 15-3.

Table 15-3
Model 4200-CVU project plans

Project Plan	Description
CVU_BJT	Measures capacitance (0 V bias) between two terminals of a bipolar junction transistor (BJT): C_{CB}, $C_{CE,}$ and C_{BE}.
CVU_Capacitor	Performs both a C-V sweep and a C-f sweep on a metal-insulator-metal (MIM) capacitor. Standard deviation is calculated.
CVU_InterconnectCap	Performs a C-V sweep for measurement of the small interconnect capacitance on a wafer.
CVU_ivcvswitch	Combines a Model 4200-SMU, Model 4200-CVU, and matrix switching in one project. Closes switches on the matrix card to connect the SMUs to a device to measure I-V on a capacitor. Then, it opens/closes another set of switches to make C-V measurements on a capacitor.
CVU_lifetime	Use to determine generation velocity and lifetime testing (Zerbst plot) of MOS capacitors. Performs both C-V and C-t sweeps and then generates a Zerbst plot, which is the generation rate plotted as a function of depletion depth.
CVU_MobileIon	Determines mobile charge using bias-temperature stress method. Makes C-V measurements on a device at room-temperature, and then repeats the C-V measurements for a hot chuck. The mobile charge is determined from the flatband shift.
CVU_MOScap	Used to measure C-V on a MOS capacitor and extract parameters, including oxide capacitance, oxide thickness, doping density, depletion depth, debye length, flatband capacitance, flatband voltage, bulk potential, threshold voltage, metal-semiconductor work function difference, and effective oxide charge.
CVU_MOSFET	Performs a two-terminal C-V sweep on a MOSFET device, and calculates oxide thickness, oxide capacitance, flatband capacitance, and flatband voltage. There is also an ITM that calculates and plots the doping concentration as a function of depletion depth.
CVU_nanowire	Performs a C-V sweep on two-terminal nanowire device to measure capacitance.
CVU_PNjunction	Measures the capacitance of a PN junction or Schottky diode as a function of the DC bias voltage across the device. Has three ITMS that include a basic C-V sweep, 1/C^2 versus V, and a doping profile.
CVU_PVcell	Measures both I-V and C-V. A SMU is used to measure the forward-biased characteristics of an illuminated solar cell and extract parameters (such as maximum power, max current, max voltage, short-circuit current, open-circuit voltage, and efficiency). The SMU also makes a reverse biased I-V measurement. C-V and C-f sweeps are performed as well.
Default	Summarizes the C-V tests that have been added to the default project. C-V testing for an NMOSFET, diode, and a capacitor.

NOTE *The basic procedure to run project plan tests is provided after the documentation for the following project plans (see Running project plan tests).*

CVU_BJT

Project Plan

Key concepts

The internal capacitance measurements on bipolar junction Transistors (BJTs) can affect the gain and frequency response of devices. The capacitance is often measured between two terminals of the device: base-emitter, base-collector, and emitter-collector. This capacitance may be measured at 0 volts or as a function of an applied voltage.

Project summary

This project has three test modules that measure the capacitance as a function of time at 0 V between the terminals of a BJT:

- C-cb0: This ITM measures the capacitance as a function of time between the collector and base at 0 V.
- C-ce0: This ITM measures the capacitance as a function of time between the collector and emitter at 0 V.
- C-be0: This ITM measures the capacitance as a function of time between the base and emitter at 0 V.

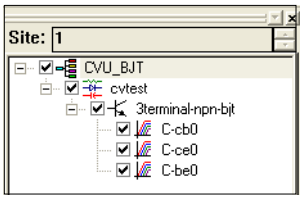
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_BJT** folder.
5. Open the **CVU_BJT** project.

The CVU_BJT project plan is shown in Figure 15-33.

Figure 15-33
CVU_BJT project plan



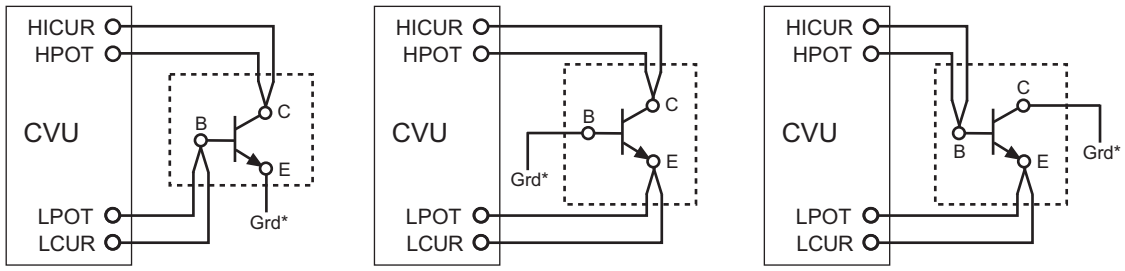
Connections

Figure 15-34 shows the basic test configurations. Note that the untested terminal must be connected to the shield of the SMA cables to guard unwanted capacitance from affecting measurement accuracy. For example, when measuring the collector-base capacitance (C-cb0), the emitter terminal is guarded by connecting it to the outer shield of the SMA cables.

Typical test connections to a DUT provides connection details. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-34
Basic configurations to test BJTs



A) C-cb0 Test B) C-ce0 Test C) C-be0 Test

* The untested terminal is to be connected to the outer shield (guard) of an SMA cable.

Formulas and constants

This project uses two formulas with no constants (see Table 15-4).

Table 15-4
Formulas for the CVU_BJT project

Formula Name	Formula Details	
AVG_CAP	Units: F	Description: Calculates the average capacitance.
	AVG_CAP = AVG(CP_AB)	
STD_DEV	Units: none	Description: Calculates the standard deviation of the capacitance measurements.
	STD_DEV = STDEV(CP_AB)	

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

C-cb0 ITM

Test summary

This ITM measures the capacitance as a function of time between the collector and base at 0 volts. The results (C versus t) are then plotted on a graph. This test also calculates the average capacitance and standard deviation.

Procedure

Before running this test, make sure the Model 4200-CVU is connected to the collector and base of the transistor (see Connections).

Parameter settings

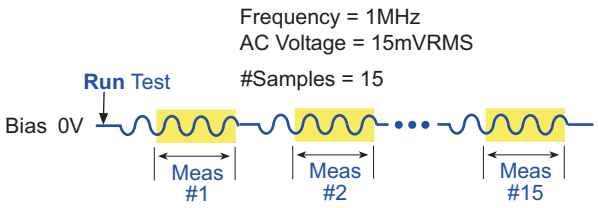
The default parameter settings for this test are listed in Table 15-5. The voltage bias waveform used for this test is shown in Figure 15-35.

Table 15-5
Parameter settings (C-cb0 ITM)

Force (Figure 15-37)	**Measure** (Figure 15-37)	**Timing** (Figure 15-36)
CVU Voltage Bias: DC Bias Conditions PreSoak: 0 V DC Bias: 0 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 15 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sampling Interval: 0.1 s #Samples: 15 Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-37)	**CVU Compensation** [3] (Figure 15-37)	
Collector: DC Source V Base: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-35
Voltage bias waveform (C-cb0 ITM)



In the Project Navigator, double-click the **C-cb0** ITM to display the Definition tab, which is shown in Figure 15-36.

Figure 15-36
Definition tab (C-cb0 ITM)

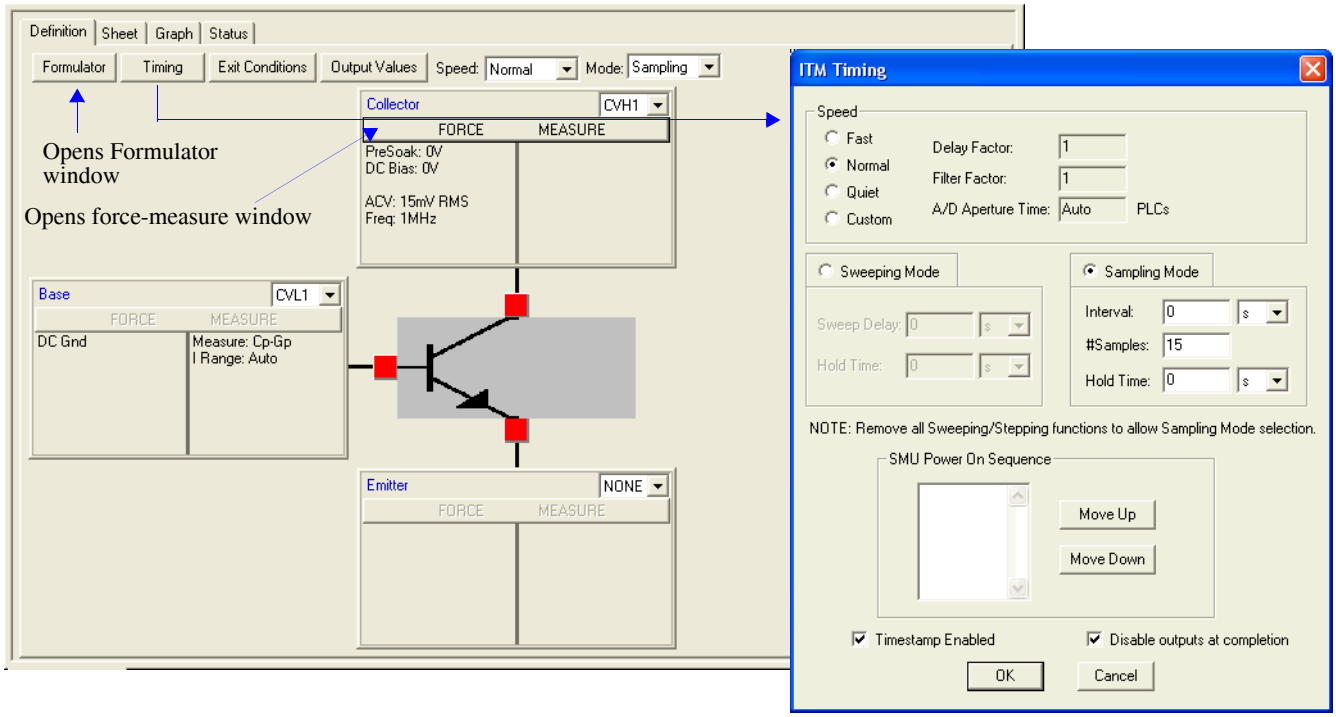
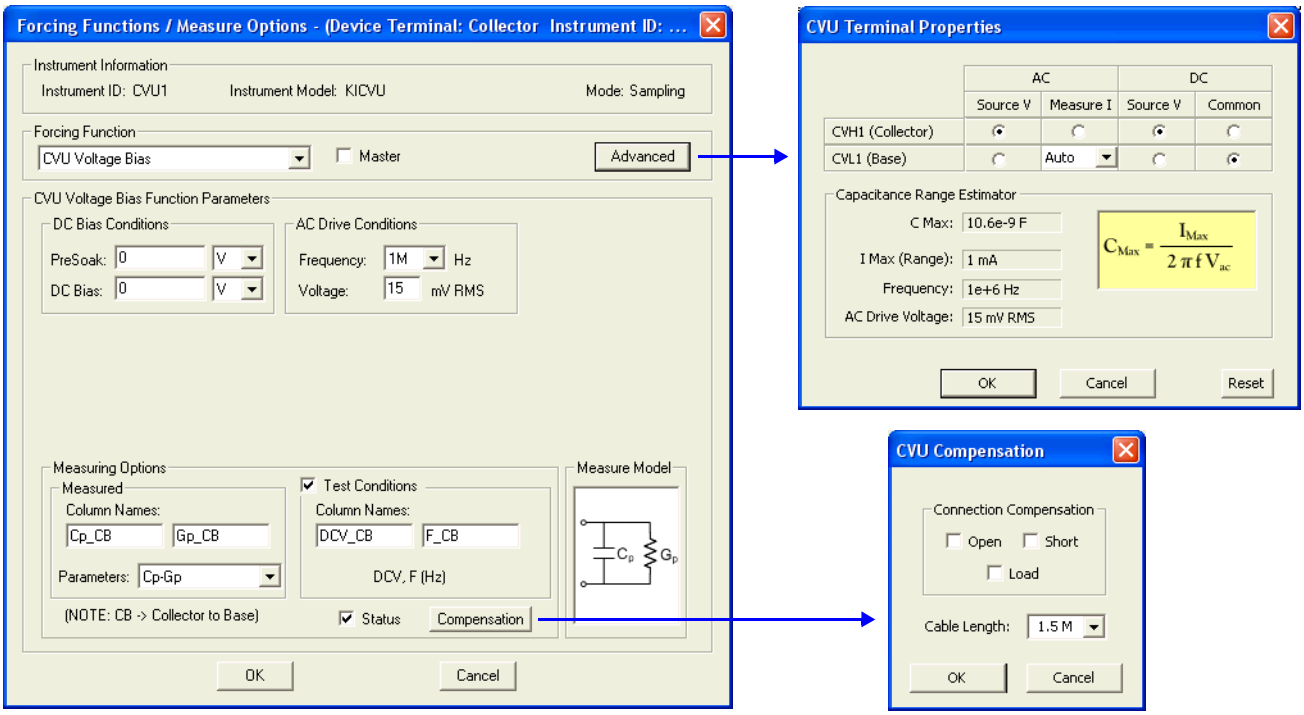


Figure 15-37
Forcing Functions / Measure Options window (C-cb0 ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

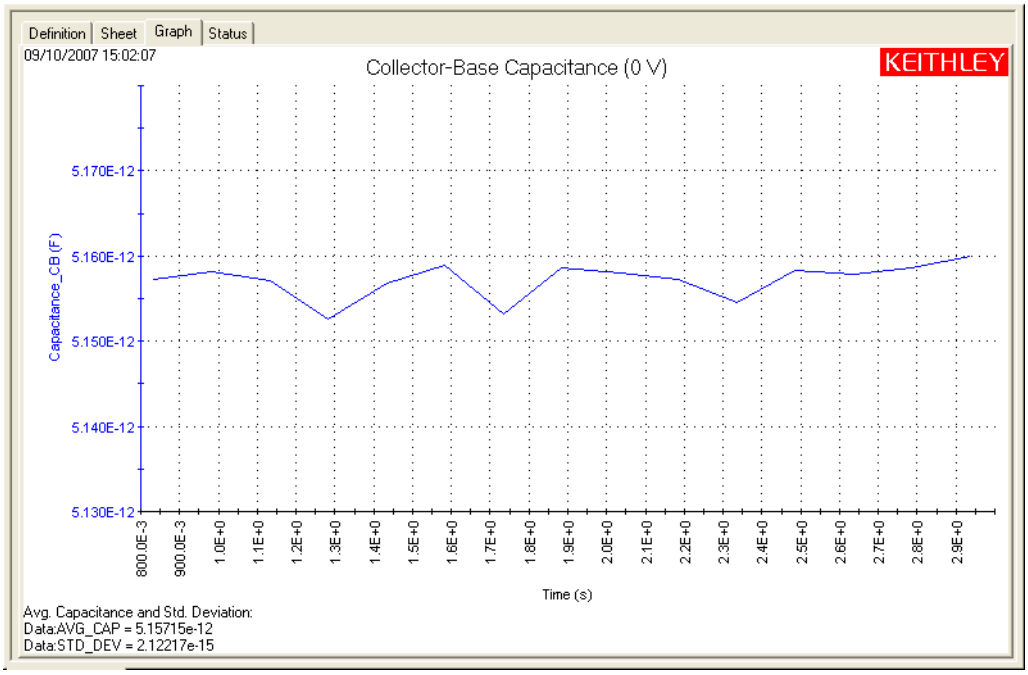
- Time Timestamp for each measurement.
- Cp_CB Measured parallel capacitance.
- Gp_CB Measured conductance.
- DCV_CB Forced DC bias voltage.
- F_CB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: CB = collector-to-base.

Graph

The graph is displayed in the Graph tab (see Figure 15-38). It includes the formulator results for average capacitance and standard deviation.

Figure 15-38
Graph tab (C-cb0 ITM)



C-ce0 ITM

Test summary

This ITM measures the capacitance as a function of time between the collector and emitter at 0 volts. The results (C versus t) are then plotted on a graph. This test also calculates the average capacitance and standard deviation.

Procedure

Before running this test, make sure the Model 4200-CVU is connected to the collector and emitter of the transistor (see Connections).

Parameter settings

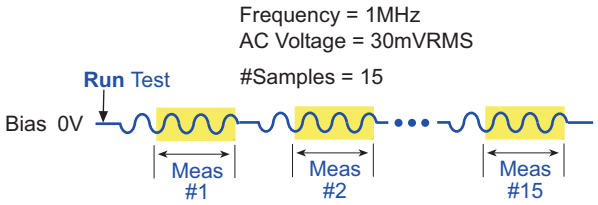
The default parameter settings for this test are listed in Table 15-6. The voltage bias waveform used for this test is shown in Figure 15-39.

Table 15-6
Parameter settings (C-ce0 ITM)

Force (Figure 15-41)	**Measure** (Figure 15-41)	**Timing** (Figure 15-40)
CVU Voltage Bias: DC Bias Conditions PreSoak: 0 V DC Bias: 0 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 30 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Quiet Mode: Sampling Interval: 0.1 s #Samples: 15 Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-41)	**CVU Compensation** [3] (Figure 15-41)	
Collector: DC Source V Emitter: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-39
Voltage bias waveform (C-ce0 ITM)



In the Project Navigator, double-click the **C-ce0** ITM to display the Definition tab, which is shown in Figure 15-40.

Figure 15-40
Definition tab (C-ce0 ITM)

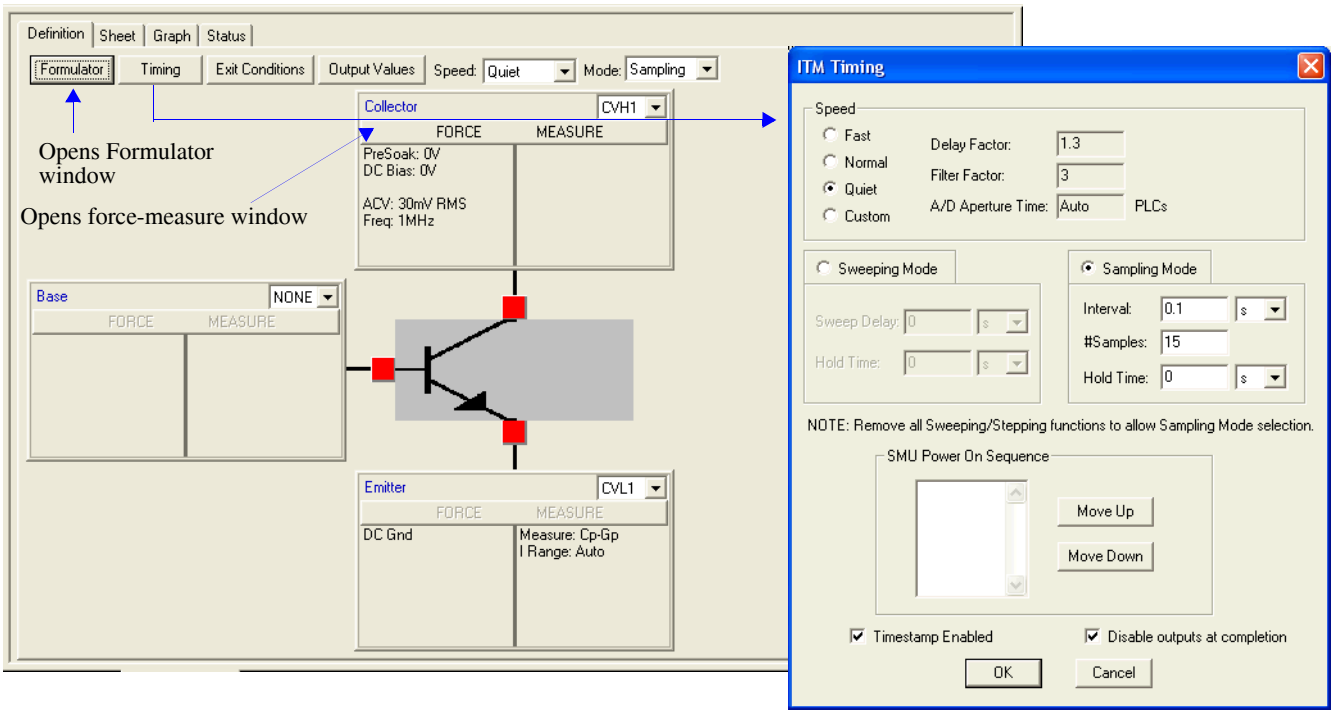
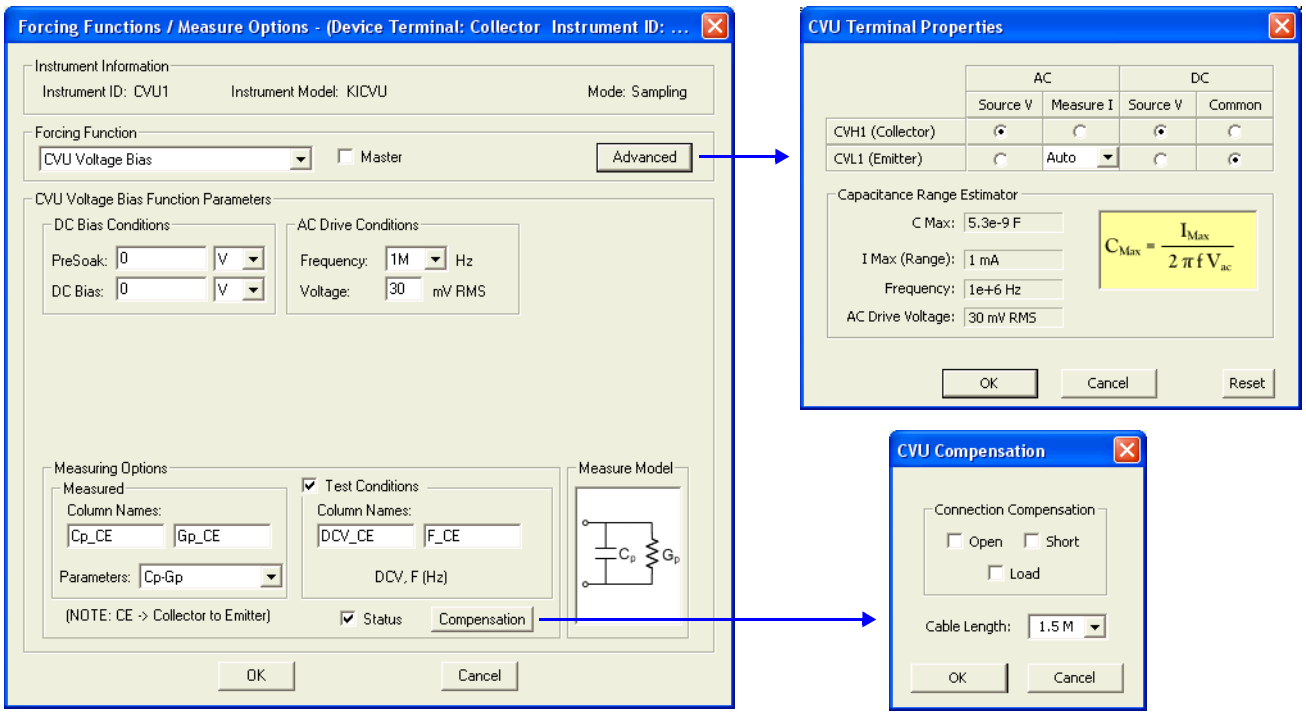


Figure 15-41
Forcing Functions / Measure Options window (C-ce0 ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

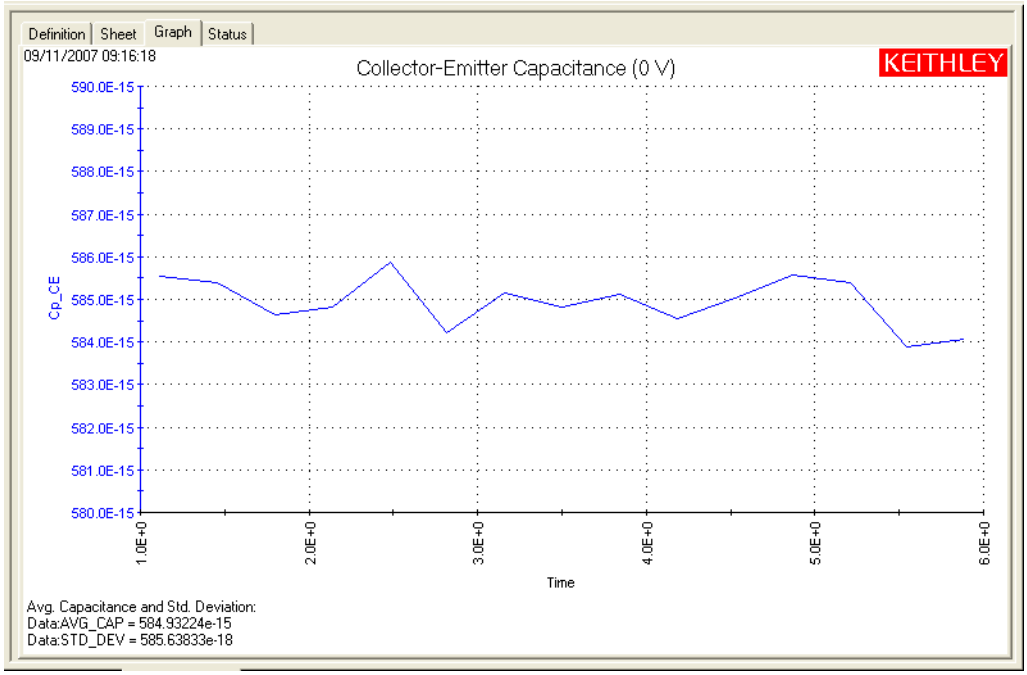
- Time Timestamp for each measurement.
- Cp_CE Measured parallel capacitance.
- Gp_CE Measured conductance.
- DCV_CE Forced DC bias voltage.
- F_CE Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: CE = collector-to-emitter.

Graph

The graph is displayed in the Graph tab (see Figure 15-42). It includes the formulator results for average capacitance and standard deviation.

Figure 15-42
Graph tab (C-ce0 ITM)



C-be0 ITM

Test summary

This ITM measures the capacitance as a function of time between the base and emitter at 0 volts. The results (C versus t) are then plotted on a graph. This test also calculates the average capacitance and standard deviation.

Procedure

Before running this test, make sure the Model 4200-CVU is connected to the base and emitter of the transistor (see Connections).

Parameter settings

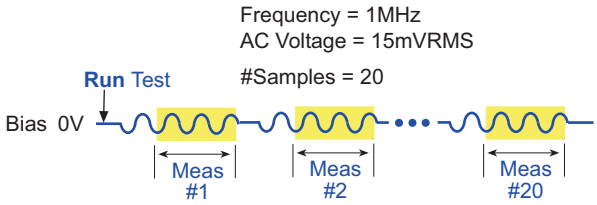
The default parameter settings for this test are listed in Table 15-7. The voltage bias waveform used for this test is shown in Figure 15-43.

Table 15-7
Parameter settings (C-be0 ITM)

Force (Figure 15-45)	**Measure** (Figure 15-45)	**Timing** (Figure 15-44)
CVU Voltage Bias: DC Bias Conditions PreSoak: 0 V DC Bias: 0 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 15 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Quiet Mode: Sampling Interval: 0.1 s #Samples: 20 Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-45)	**CVU Compensation** [3] (Figure 15-45)	
Collector: DC Source V Emitter: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-43
Voltage bias waveform (C-be0 ITM)



In the Project Navigator, double-click the **C-be0** ITM to display the Definition tab, which is shown in Figure 15-44.

Figure 15-44
Definition tab (C-be0 ITM)

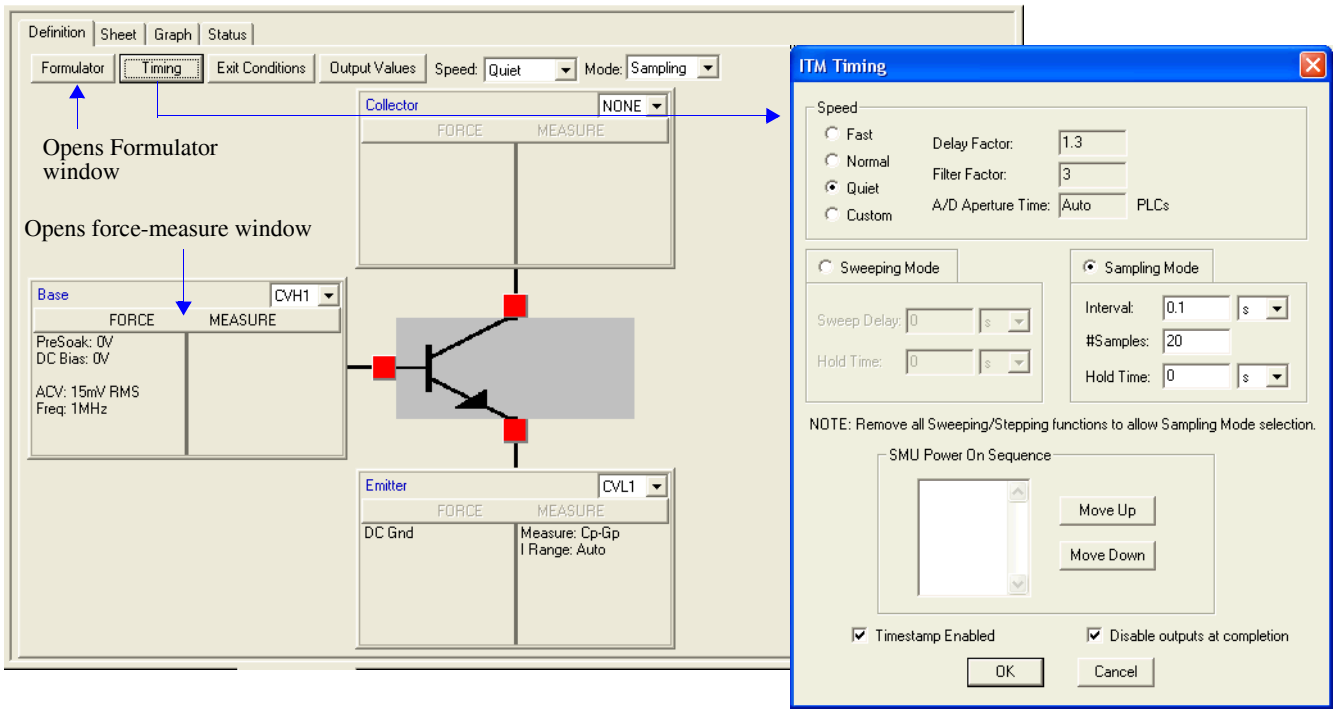
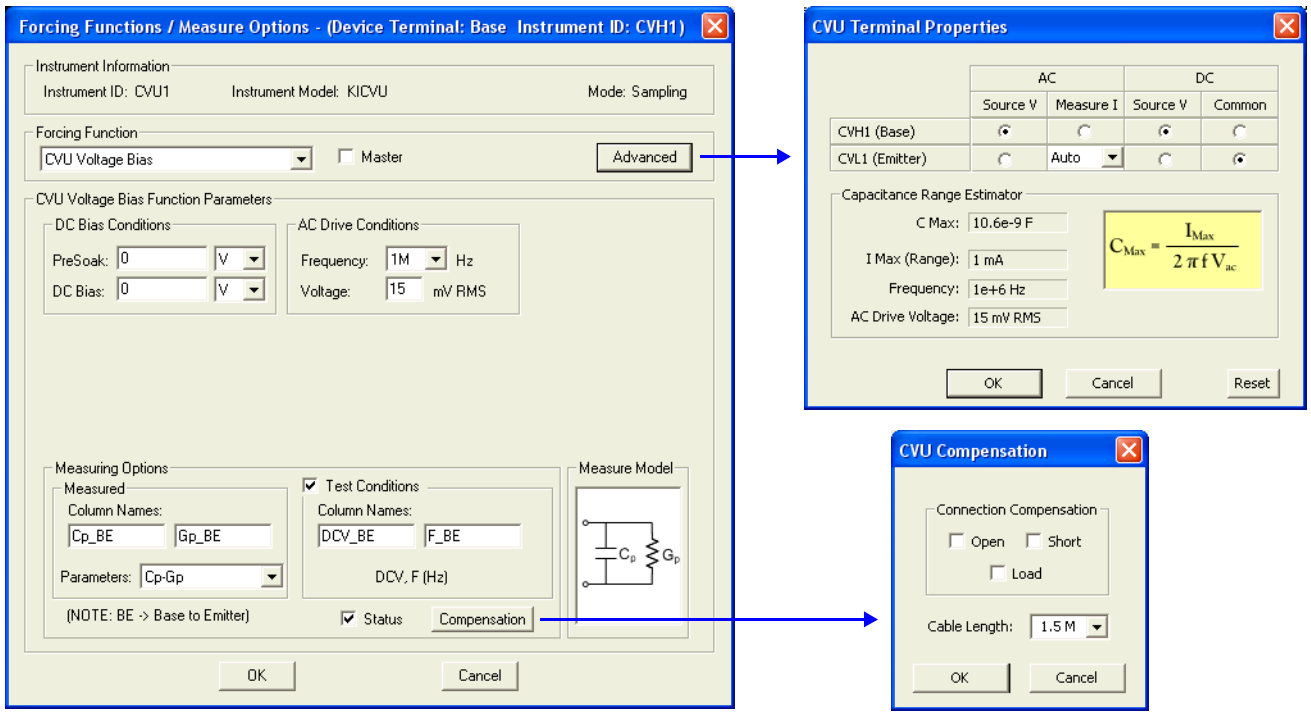


Figure 15-45
Forcing Functions / Measure Options window (C-be0 ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

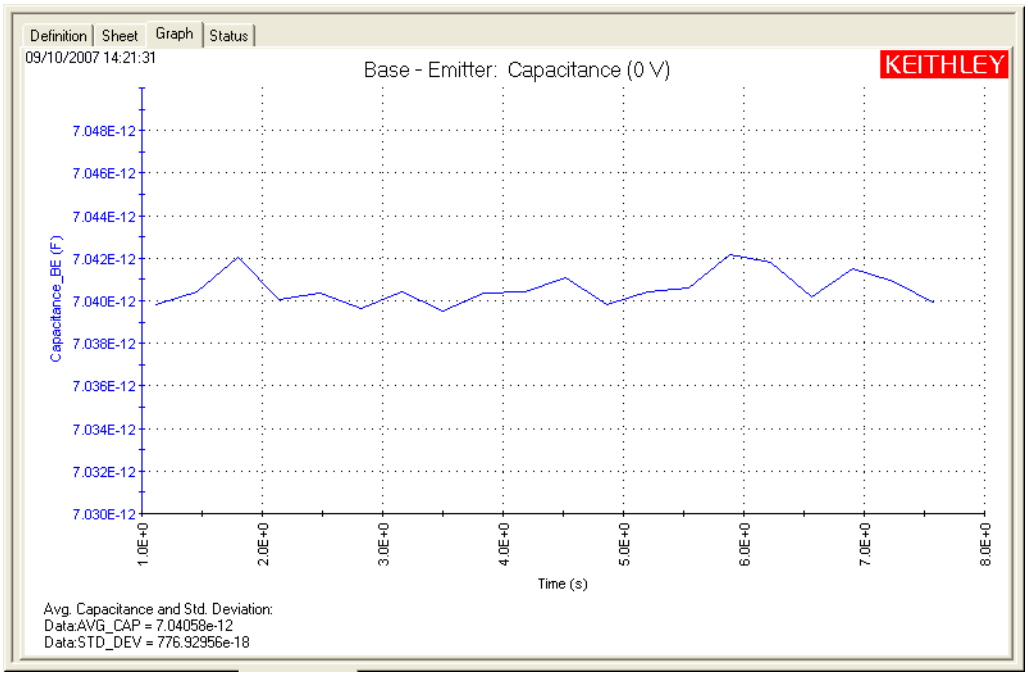
- Time Timestamp for each measurement.
- CPA Measured parallel capacitance.
- Gape Measured conductance.
- DCV_BE Forced DC bias voltage.
- F_BE Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: BE = base-to-emitter.

Graph

The graph is displayed in the Graph tab (see Figure 15-46). It includes the formulator results for average capacitance and standard deviation.

Figure 15-46
Graph tab (C-be0 ITM)



CVU_Capacitor

Project Plan

Project summary

This project performs both a capacitance-voltage sweep and a capacitance-frequency sweep on a Metal-Insulator-Metal (MIM) capacitor (10 pF). The following graphs are generated:

- C versus V Using a voltage sweep, capacitance is measured at every step of the sweep to generate a Capacitance versus Voltage graph. Noise is also calculated.
- C versus F Using a frequency sweep, capacitance is measured at every frequency point to generate a Capacitance versus Frequency graph.

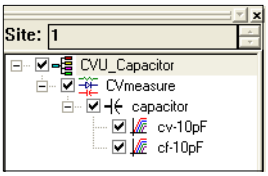
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_Capacitor** folder.
5. Open the **CVU_Capacitor.kpr** project.

The CVU_Capacitor project plan is shown in Figure 15-47.

Figure 15-47
CVU_Capacitor project plan

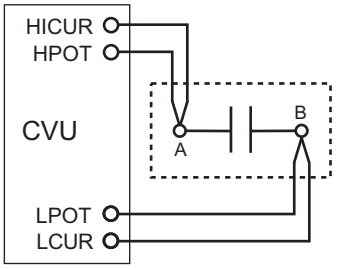


Connections

The test configuration is shown in Figure 15-48. For details on connections, see Connections. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-48
Basic configuration to test a MIM capacitor



Formulas and constants

This project uses one formula with no constants (see Table 15-8, below).

Table 15-8
Formula for the CVU_Capacitor project

Formula Name	Formula Details	
NOISE	Units: F	Description: Calculates the standard deviation of the capacitance measurements.
	NOISE = STDEV(CP_AB)	

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

cv-10 pF ITM

Test summary

This ITM performs a voltage sweep measuring capacitance on each step, and generates a C versus V graph. It also calculates noise.

Parameter settings

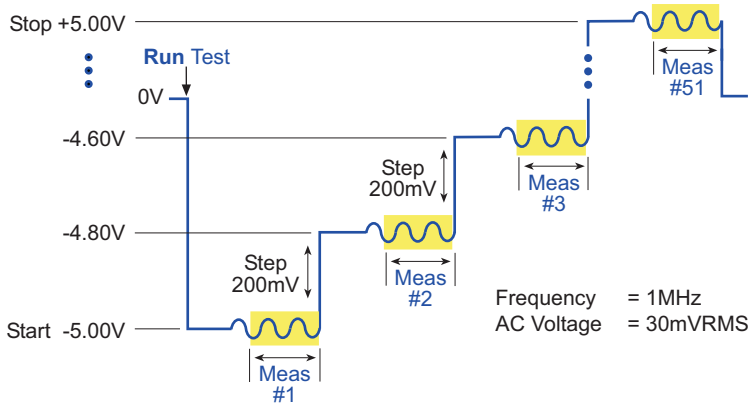
The default parameter settings for this test are listed in Table 15-9. The voltage sweep used for this test is shown in Figure 15-49.

Table 15-9
Parameter settings (cv-10 pF ITM)

Force (Figure 15-51)	**Measure** (Figure 15-51)	**Timing** (Figure 15-50)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -5 V Stop: 5 V Step: 0.2 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 30 mV Data Points: 51	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-51)	**CVU Compensation** [3] (Figure 15-51)	
Terminal A: DC Source V Terminal B: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-49
Voltage sweep (cv-10 pF ITM)



In the Project Navigator, double-click the **cv-10 pF** ITM to display the Definition tab, which is shown in Figure 15-50.

Figure 15-50
Definition tab (cv-10 pF ITM)

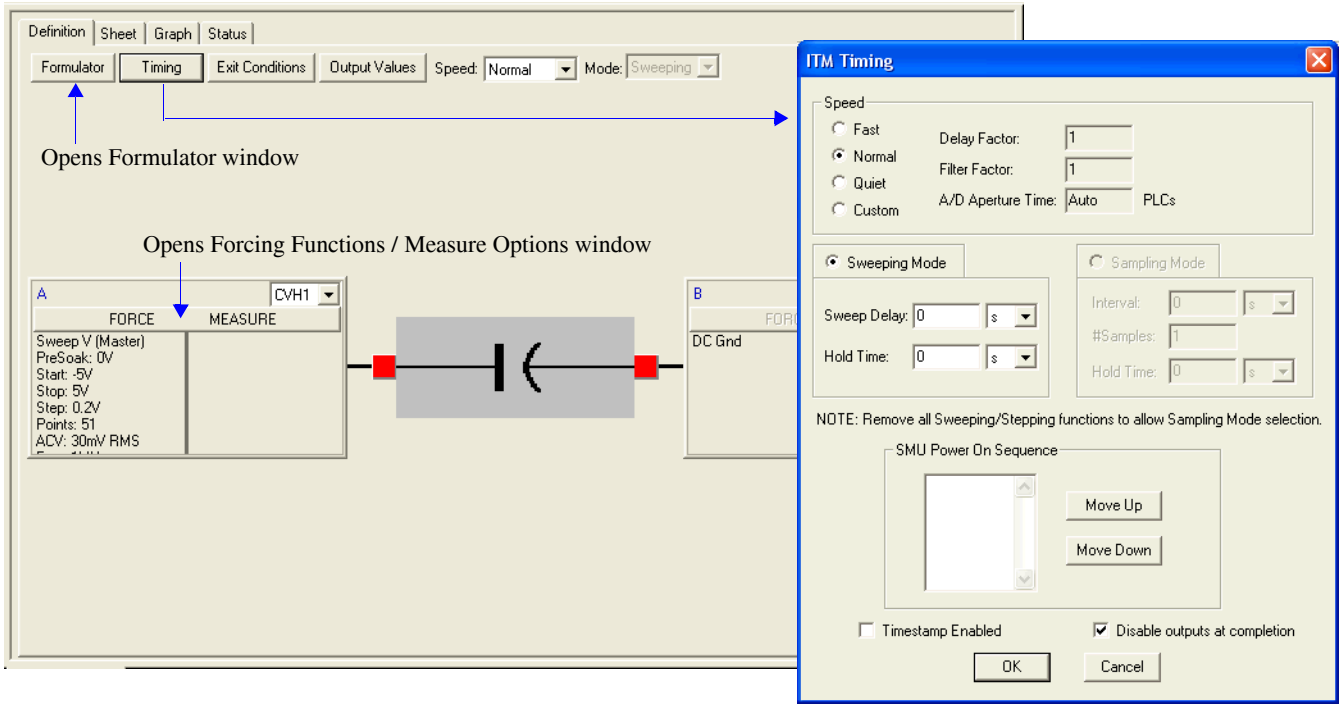
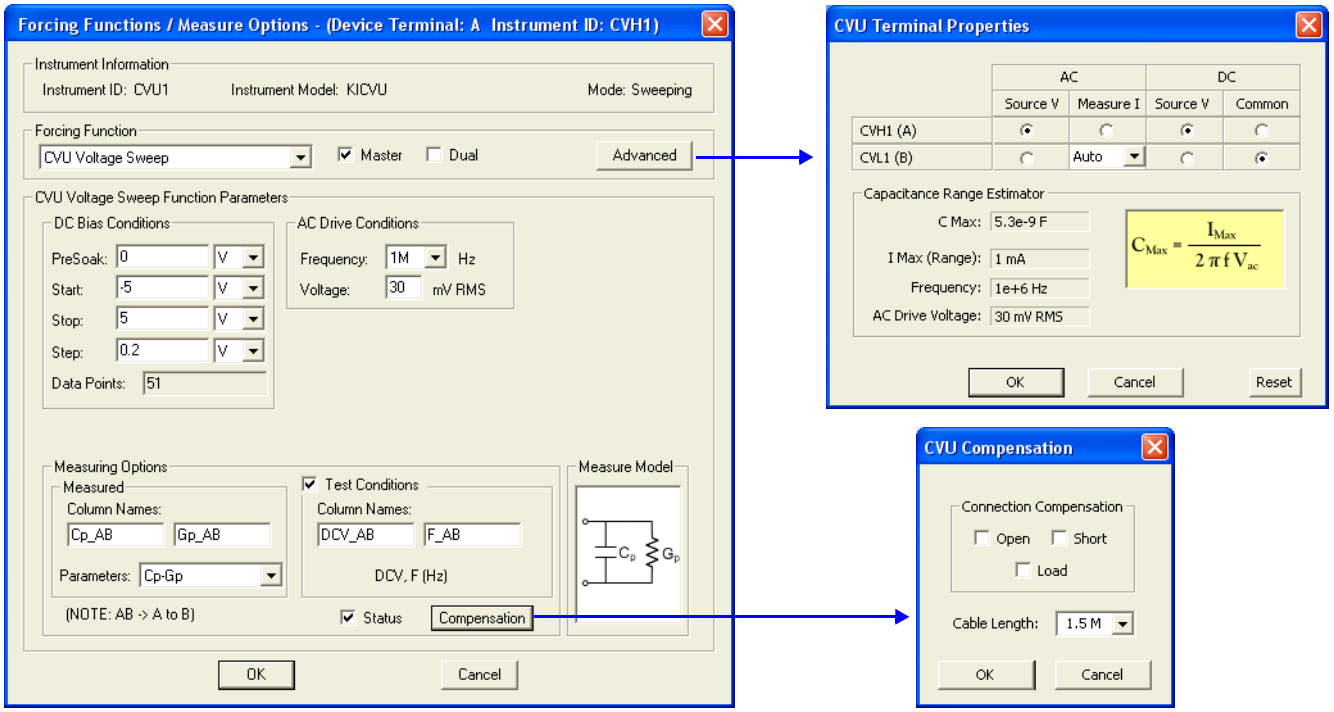


Figure 15-51
Forcing Functions / Measure Options window (cv-10 pF ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Cp_AB Measured parallel capacitance.
- Gp_AB Measured conductance.

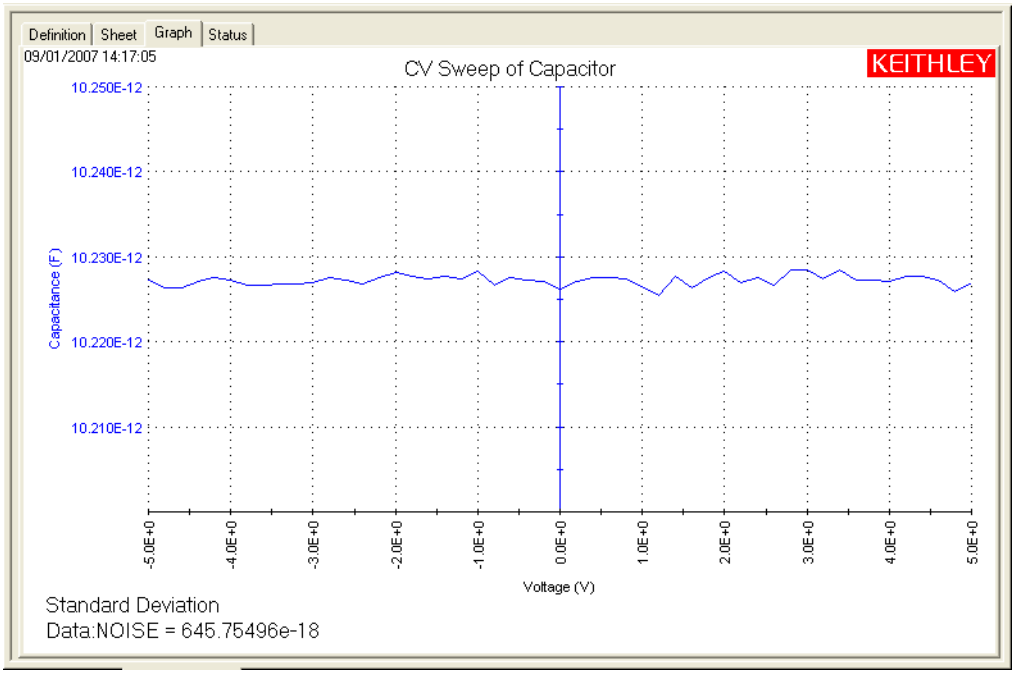
- DCV_AB Forced DC bias voltage.
- F_AB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- NOISE Formulator calculation result.

Note: AB = Terminal A to Terminal B.

Graph

The graph is displayed in the Graph tab (see Figure 15-52). It includes the formulator result for noise.

Figure 15-52
Graph tab (cv-10 pF ITM)



cf-10 pF ITM

Test summary

This ITM performs a frequency sweep measuring capacitance at each frequency point, and generates a C versus F graph.

Parameter settings

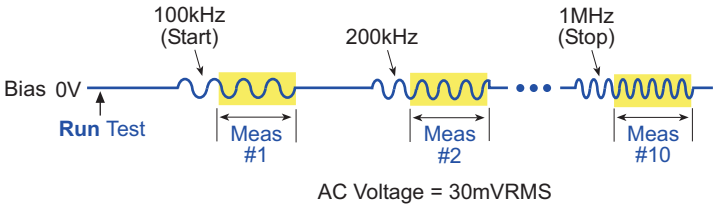
The default parameter settings for this test are listed in Table 15-10. The frequency sweep used for this test is shown in Figure 15-53.

Table 15-10
Parameter settings (cf-10 pF ITM)

Force (Figure 15-55)	**Measure** (Figure 15-55)	**Timing** (Figure 15-54)
CVU Frequency Sweep: DC Bias Conditions PreSoak: 0 V DC Bias: 0 V AC Drive Conditions Start Frequency: 100 kHz Stop Frequency: 1 MHz AC Voltage: 30 mV Data Points: 10	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-55)	**CVU Compensation** [3] (Figure 15-55)	
Terminal A: DC Source V Terminal B: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-53
Frequency sweep (cf-10 pF ITM)



In the Project Navigator, double-click the **cf-10 pF** ITM to display the Definition tab, which is shown in Figure 15-54.

Figure 15-54
Definition tab (cf-10 pF ITM)

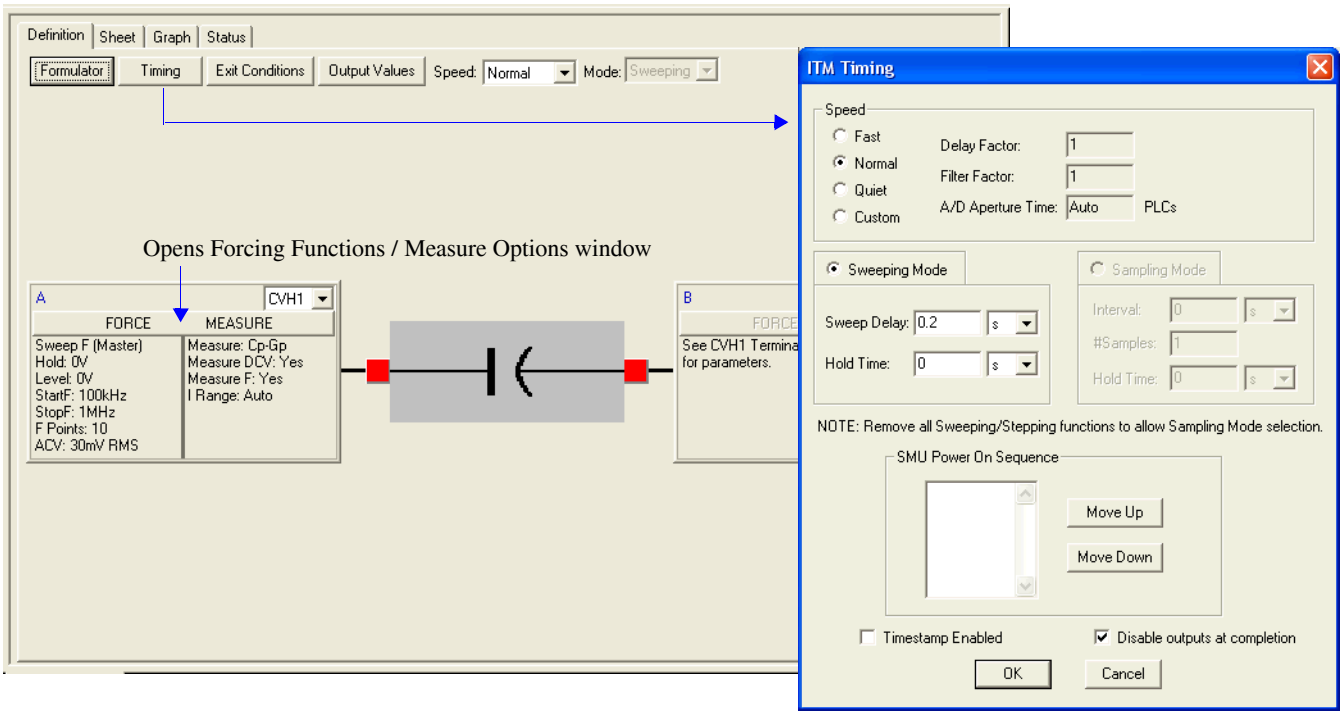
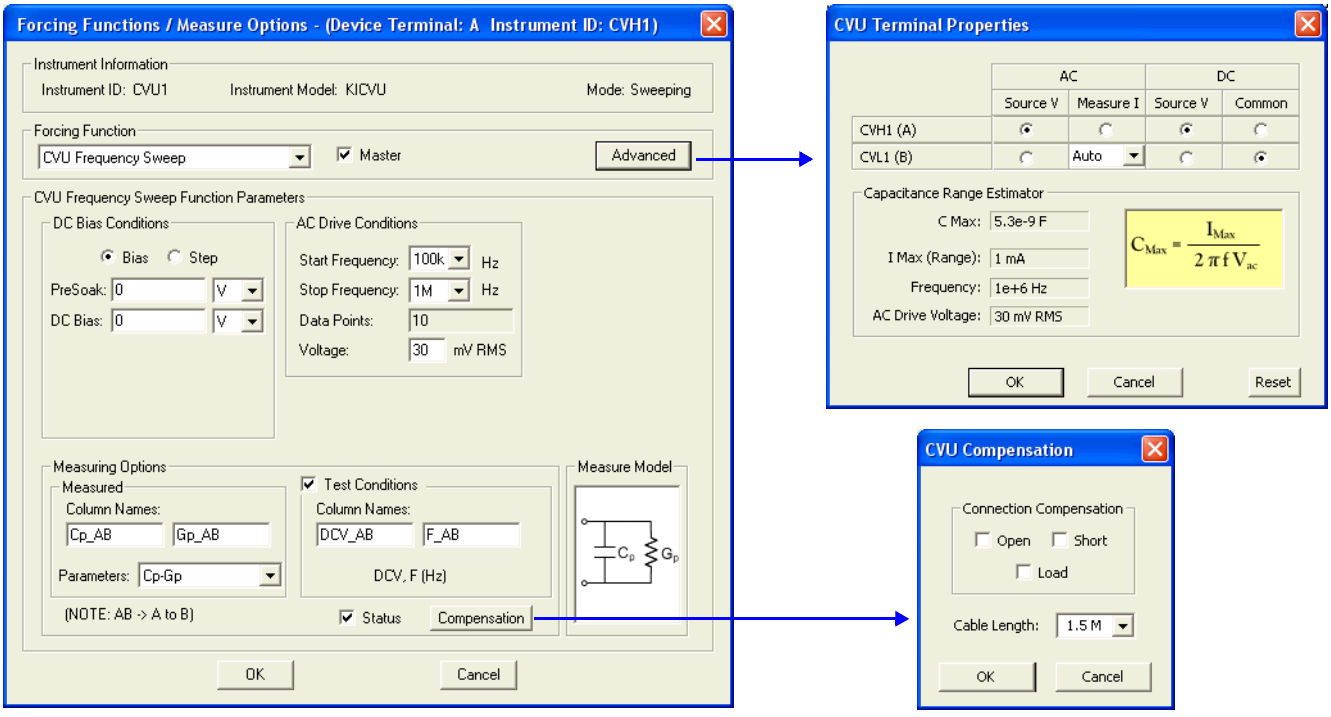


Figure 15-55
Forcing Functions / Measure Options window (cf-10 pF ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Cp_AB Measured parallel capacitance.
- Gp_AB Measured conductance.

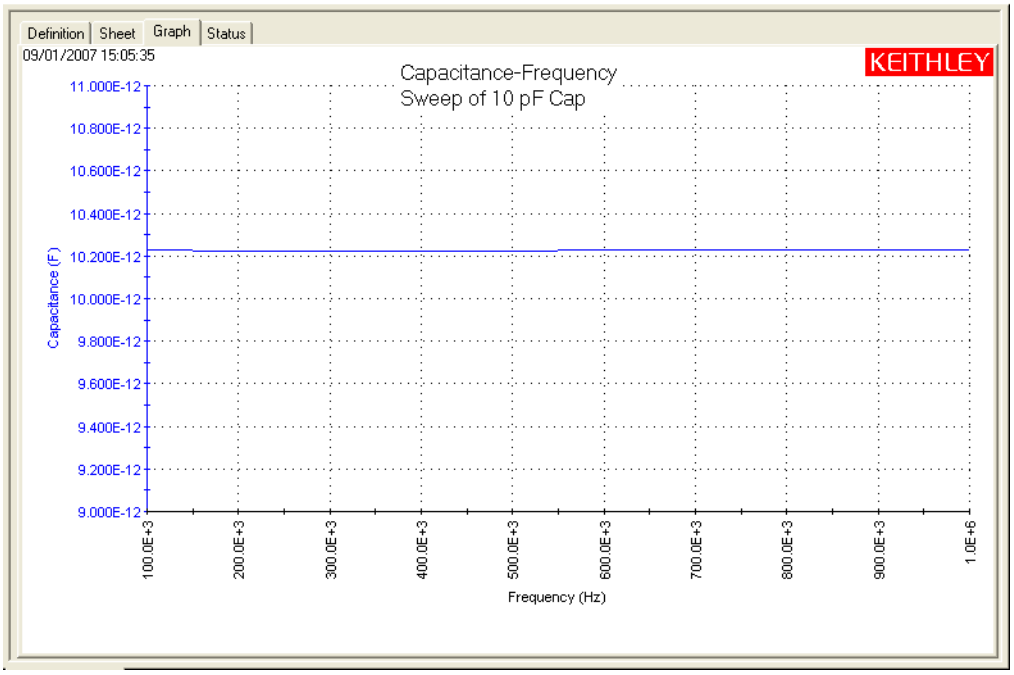
- DCV_AB Forced DC bias voltage.
- F_AB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.

Note: AB = Terminal A to Terminal B.

Graph

The graphical results for the Capacitance versus Frequency sweep test is displayed in Figure 15-56.

Figure 15-56
Graph tab (cf-10 pF ITM)



CVU_InterconnectCap

Project Plan

Key concepts

Because the interconnect capacitance directly affects the speed and noise of an integrated circuit, making accurate capacitance measurements is very important. These measurements are usually measured between two metal pads on the wafer. The magnitude of capacitance is usually very small (<1pF).

Besides the capacitance measurements, I-V measurements can made between the metal pads as well. An ITM can be inserted into this project to measure the leakage current between the wafer pads using Model 4200-SMUs.

Project summary

This project uses a voltage sweep to measure capacitance at every step of the sweep and generates a capacitance versus voltage graph.

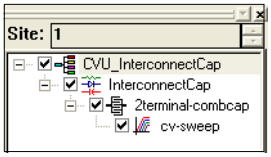
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the _CV folder.
4. Double-click the **CVU_InterconnectCap** folder.
5. Open the **CVU_InterconnectCap.kpr** project.

The CVU_InterconnectCap project plan is shown in Figure 15-57. As shown, the project consists of one ITM.

Figure 15-57
CVU_InterconnectCap project plan

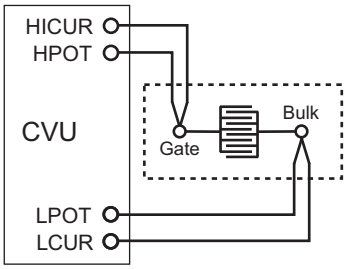


Connections

The test configuration in Figure 15-58 shows the schematic representation of interconnect capacitance on a wafer. For details on connections to the wafer, see Connections. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-58
Basic configuration to test interconnect capacitance



Formulas and constants

This project uses one formula (no constants) (see Table 15-11).

Table 15-11
Formula for the CVU_InterconnectCap project

Formula Name	Formula Details	
NOISE	Units: F	Description: Calculates the standard deviation of the capacitance measurements.
	NOISE = STDEV(CP_GB)	

Running project plan tests

NOTE *Before running the project, an correction should be performed and enabled to compensate for errors due to the switching matrix (see Connection compensation). After performing connection compensation, Open correction must be enabled from the CVU Compensation window shown in Figure 15-61.*

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

cv-sweep ITM

Test summary

This ITM performs a voltage sweep measuring capacitance on each step, generates a C versus V graph and calculates noise.

Parameter settings

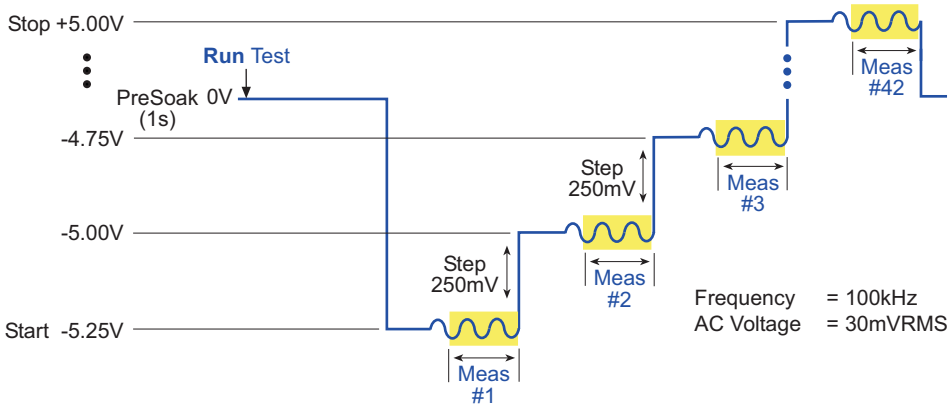
The default parameter settings for this test are listed in Table 15-12. The voltage sweep used for this test is shown in Figure 15-59.

Table 15-12
Parameter settings (cv-sweep ITM)

Force (Figure 15-61)	**Measure** (Figure 15-61)	**Timing** (Figure 15-60)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -5.25 V Stop: 5 V Step: 0.25 V AC Drive Conditions Frequency: 100 kHz AC Voltage: 30 mV Data Points: 42	Cp-Gp DCV Frequency I Range: 1µA [1] Status: Enabled	Speed: Quiet Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 1 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-61)	**CVU Compensation** [3] (Figure 15-61)	
Gate: DC Source V Bulk: AC Measure I (Auto)	Enable Open compensation for this test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-59
Voltage sweep (cv-sweep ITM)



In the Project Navigator, double-click the **cv-sweep** ITM to display the Definition tab, which is shown in Figure 15-60.

Figure 15-60
Definition tab (cv-sweep ITM)

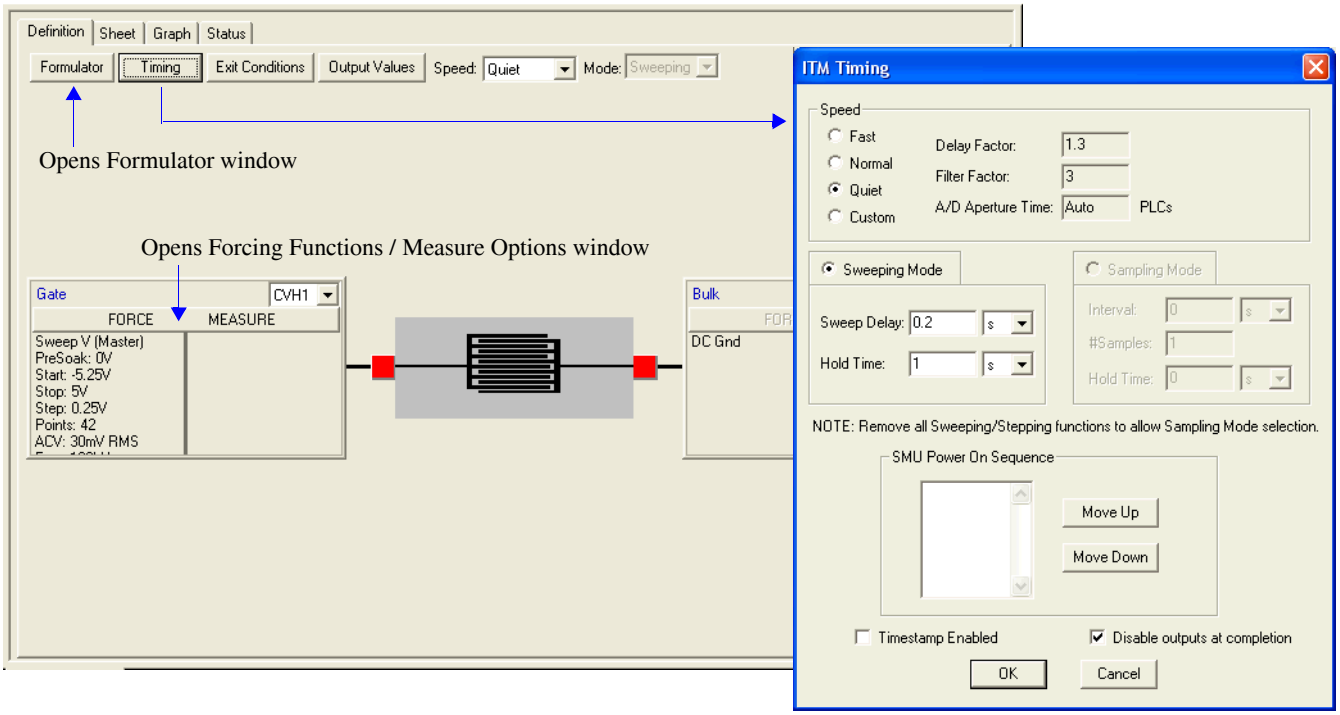
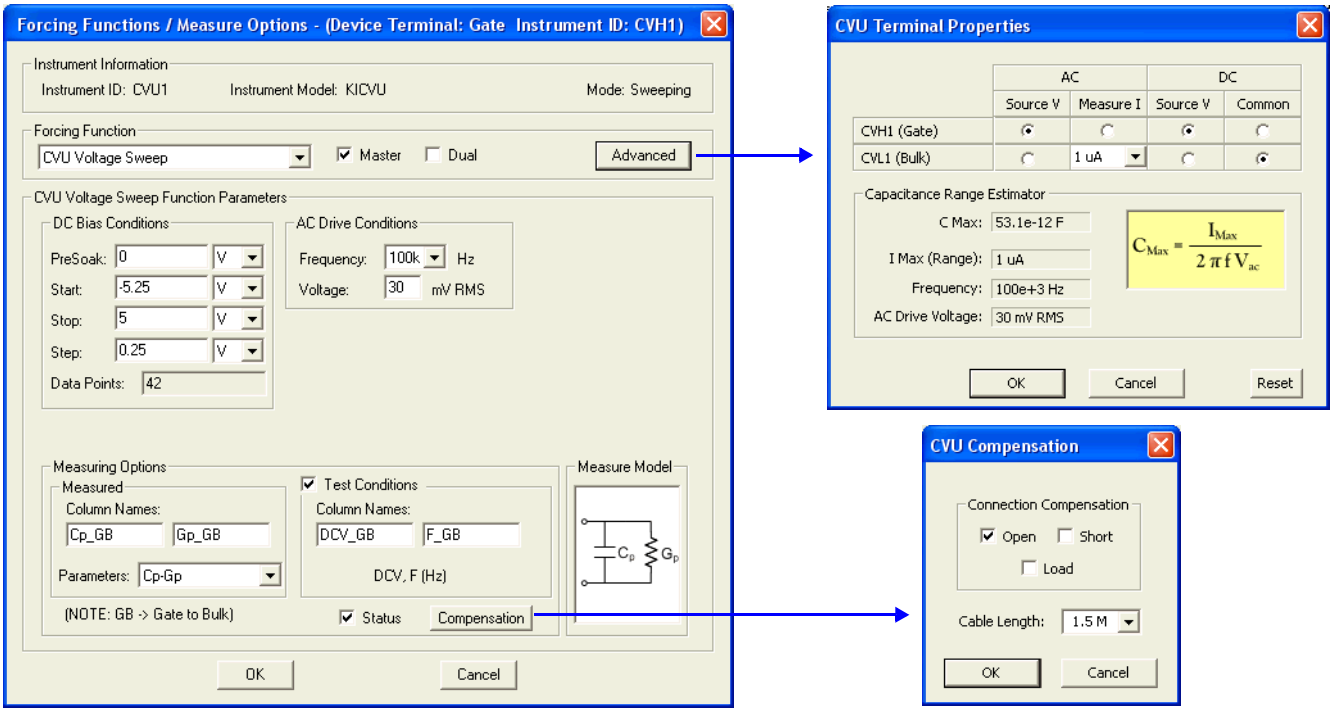


Figure 15-61
Forcing Functions / Measure Options window (cv-sweep ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

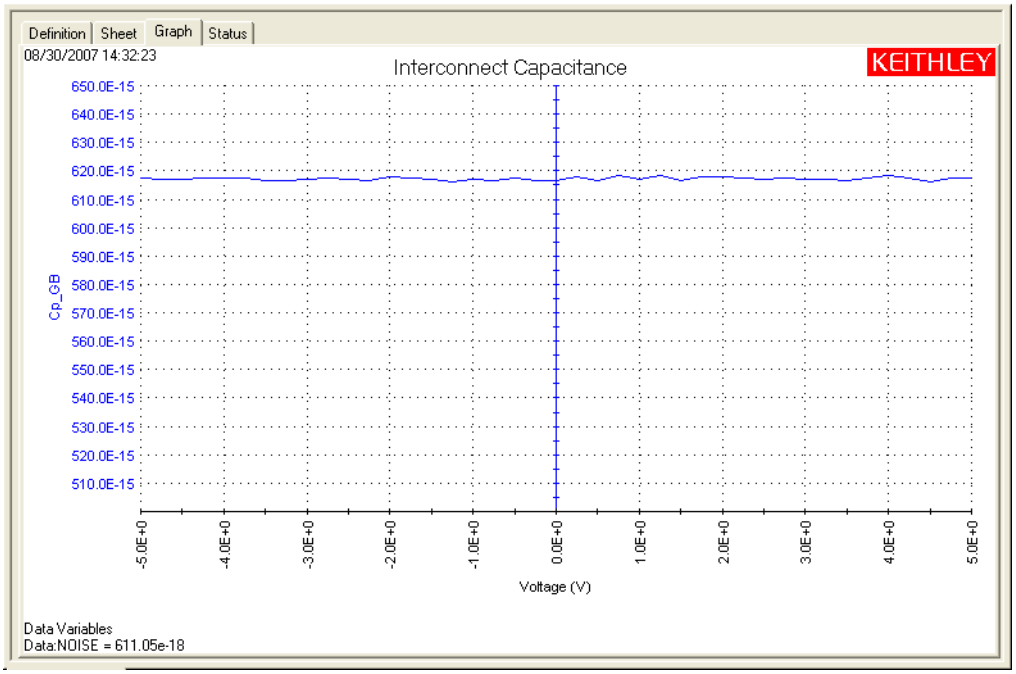
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- NOISE Formulator calculation result.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-62). It includes the formulator result for noise.

Figure 15-62
Graph tab (cv-sweep ITM)



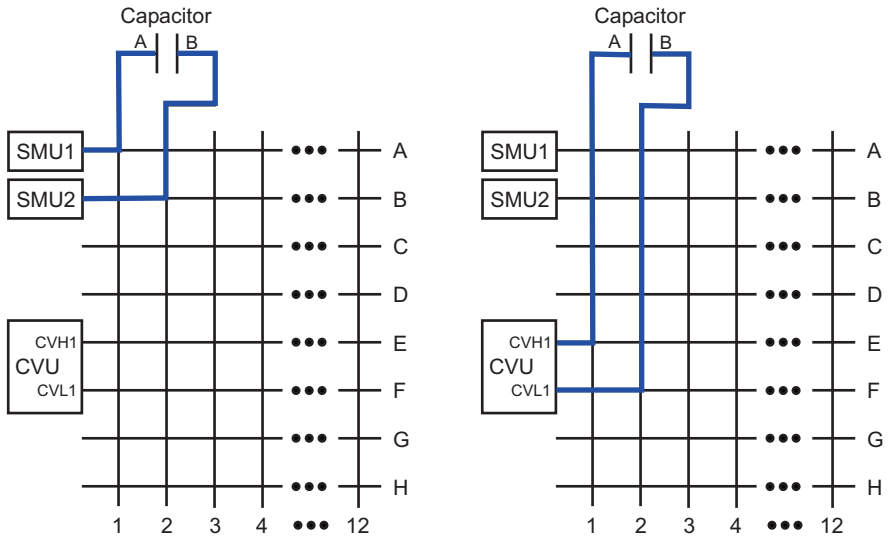
CVU_ivcvswitch

Project Plan

Key concepts

This project demonstrates how a switching matrix can be used to automate I-V and C-V testing of a capacitor. When the project is run, the switching matrix will connect two Model 4200-SMUs to the capacitor (see Figure 15-63A) and I-V testing will be performed. The switching matrix will then connect the Model 4200-CVU to the capacitor (see Figure 15-63B) and C-V testing will be performed.

Figure 15-63
Switching matrix signal paths



A) Signal Path for I-V Testing B) Signal Path for C-V Testing

The switching matrix required for this project is the Model 707A or 708A switching mainframe and a Model 7174A matrix card. The 707A/708A Mainframes and 7174A Matrix Card must first be configured in KCON (see Adding and configuring a switching matrix below).

NOTE *The Model 7072 matrix card can also be used for this project plan. However, you must use Rows G and H.*

Adding and configuring a switching matrix

KCON system configuration is used to add the switching matrix to the Model 4200-SCS test system and configure it.

In KCON (Keithley Configuration Utility), use the Properties tab for the switching matrix and the Properties tab for the matrix card to add and configure the switching matrix. Make sure that sensing is set for local (2-wire). Figure 15-27 and Figure 15-28 show the required KCON connection settings for the switching mainframe and the matrix card.

Project summary

This project tests a capacitor by measuring leakage current (using a Model 4200-SMU) and capacitance (using the Model 4200-CVU). The following graphs are generated by these tests:

Current versus Time and Capacitance versus Time. When run from the project plan level, the testing sequence proceeds as follows:

1. All switches for the matrix are opened.
2. Switches for the matrix card are closed to connect the SMUs to a capacitor (see Figure 15-63A). The SMU sources a fixed bias voltage to charge the capacitor, and measures leakage current as a function of time.
3. Switches are opened to disconnect the SMUs, and then closed to connect the CVU to the capacitor (see Figure 15-63B).
4. The CVU sources a fixed bias voltage to the capacitor and measures capacitance as a function of time. The average capacitance and standard deviation are calculated and displayed on the C-t graph.

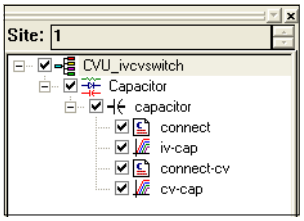
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_ivcvswitch** folder.
5. Open the **CVU_ivcvswitch.kpr** project.

The CVU_ivcvswitch project plan is shown in Figure 15-64. As shown, the project consists of two UTMs and two ITMs.

Figure 15-64
CVU_ivcvswitch project plan



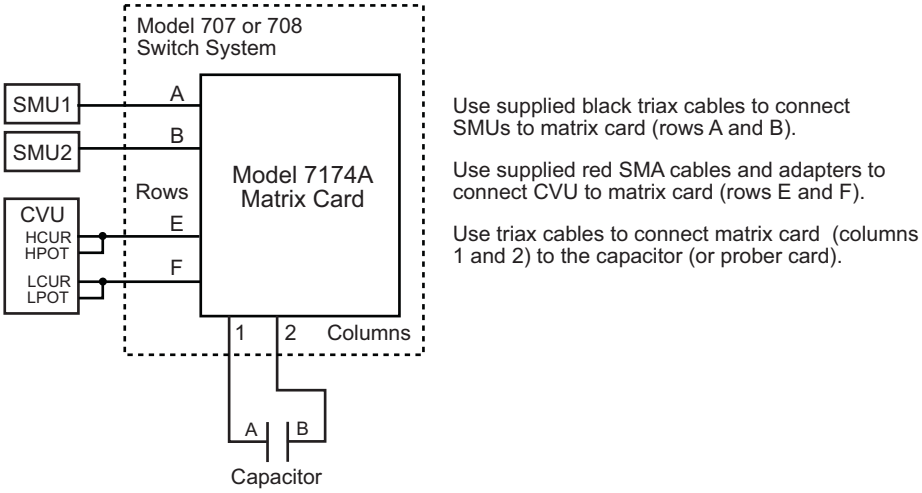
Connections

Figure 15-65 shows the basic test configuration: details on Model 4200-CVU connections are provided in Connections. See Section 4 for details on Model 4200-SMU connections.

Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to perform the Confidence Check and Connection compensation (Open and Short correction).*

Figure 15-65
Simplified configuration to use a switching matrix for I-V and C-V testing



NOTE *Details on CVU connections to the matrix card are provided in Figure 15-12.*

Formulas and constants

This project uses two formulas (with no constants), that are shown in Table 15-13.

Table 15-13
Formula for the CVU_ivcvswitch

Formula Name	Formula Details	
AVG_CAP	Units: F	Description: Calculates the average capacitance.
	AVG_CAP = AVG(CP_AB)	
STD_DEV	Units: none	Description: Calculates the standard deviation of the capacitance measurements.
	STD_DEV = STDEV(CP_AB)	

Running project plan tests

NOTE *Before running the project, an Open and Short correction should be performed and enabled to compensate for errors due to the switching matrix (see Connection compensation). After performing connection compensation, Open and Short correction must be enabled from the CVU Compensation window shown in Figure 15-73.*

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

connect and **connect-cv UTMs**

Test summaries

These two User Test Modules (UTMs) are used to control the matrix card switches. The first UTM (connect) is used to connect the Model 4200-SMUs to the capacitor. The second UTM (connect-cv) is used to disconnect the SMUs, and then connect the Model 4200-CVU to capacitor. The Project summary explains the test sequence.

Definition tabs

The Definition tab for a UTM is opened by double-clicking the UTM in the project navigator. Figure 15-66 shows the Definition tab for the connect UTM, and Figure 15-67 shows the Definition tab for the connect-cv UTM.

connect UTM:

- The OpenAll parameter (line 1) is set for 1. This opens all matrix switches at the beginning of the test sequence.
- The SMU1 parameter (lines 2 and 3) is set for pin 1. This connects SMU1 to column 1 of the matrix card. As shown in Figure 15-65, column 1 is connected to terminal A of the capacitor.
- The SMU2 parameter (lines 4 and 5) is set for pin 2. This connects SMU2 to column 2 of the matrix card. Column 2 is connected to terminal B of the capacitor.
- All the other instrument parameters are set for 0 (not used).

connect-cv UTM:

- The OpenAll parameter (line 1) is set for 1. This opens all switches after the iv-cap test is finished.
- The CVH1 parameter (lines 10 and 11) is set for pin 1. This connects CVH1 to column 1 of the matrix card. Column 1 is connected to terminal A of the capacitor.
- The CVL1 parameter (lines 12 and 13) is set for pin 2. This connects CVL1 to column 2 of the matrix card. Column 2 is connected to terminal B of the capacitor.
- All the other instrument parameters are set for 0 (not used).

Figure 15-66
Definition tab (connect ITM)

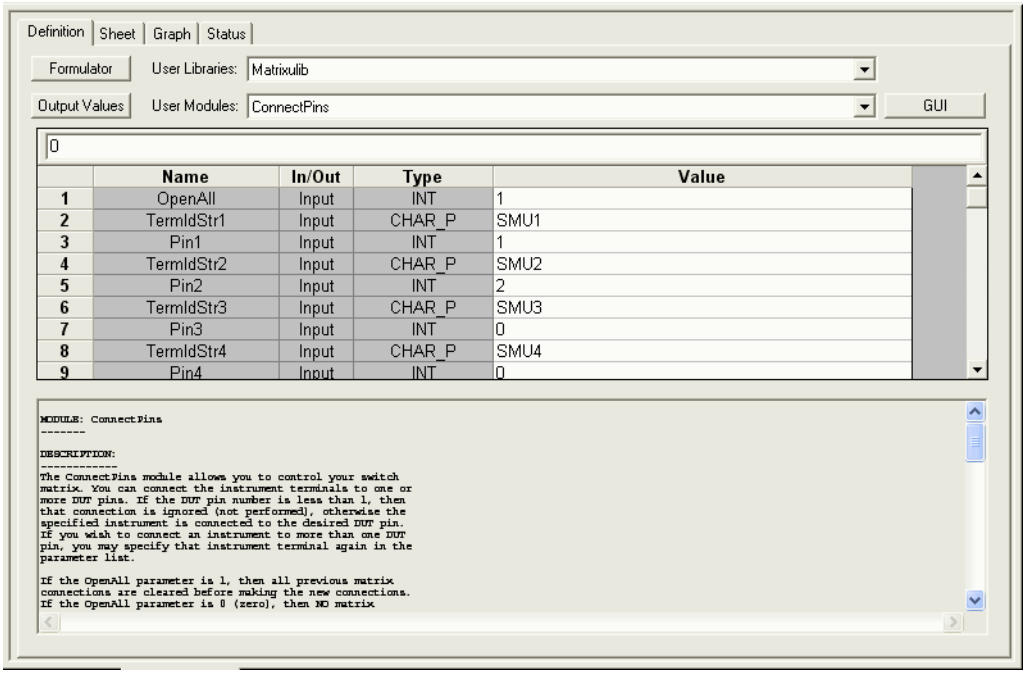
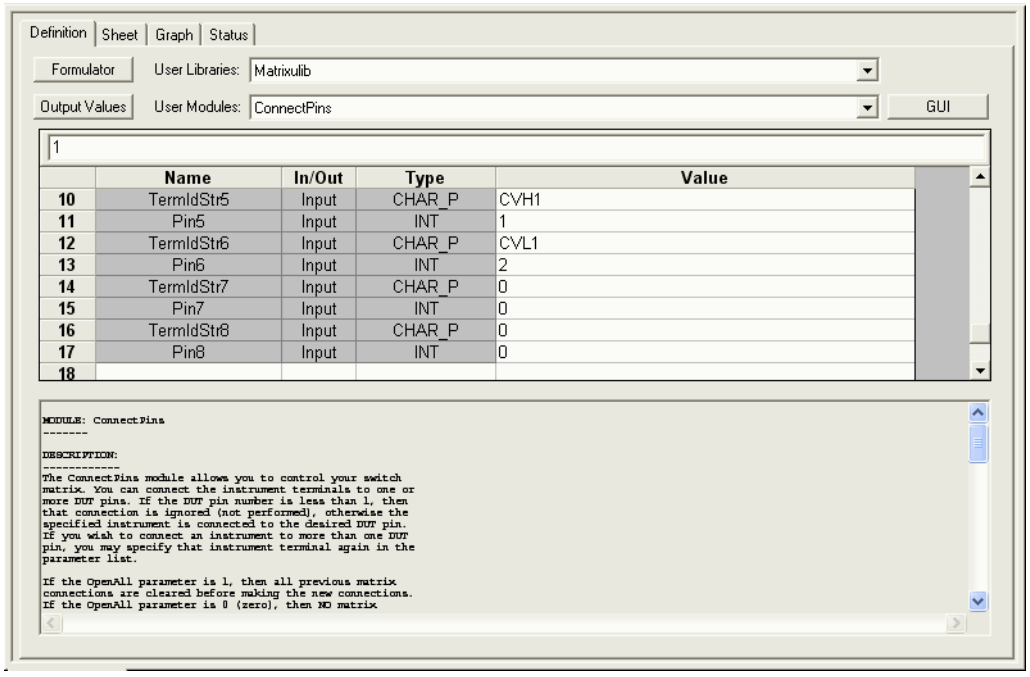


Figure 15-67
Definition tab (connect-cv ITM)



iv-cap UTM

Test summary

After the connect User Test Module (UTM) connects the Model 4200-SMUs to the capacitor, SMU1 sources a fixed DC bias voltage to charge the capacitor, and measures leakage current as a function of time.

Force, measure, and timing settings

The parameter settings for the two Model 4200-SMUs are listed in Table 15-14:

- SMU1 is configured to bias 5 V and perform 60 current measurements (at 100 ms intervals).
- SMU2 is configured as a Common for the return signal path.

In the Project Navigator, double-click the **iv-cap** ITM to display the Definition tab. From the Definition tab, parameters are set from the ITM Timing window and the Forcing Functions / Measure Options windows. Figure 15-69 shows the force-measure window for SMU1. The force-measure window for SMU2 (not shown) is configured as a Common.

Table 15-14
Parameter settings

Force	Measure	Timing
SMU1: Forcing Function: Voltage Bias Level: 5 V Compliance: 0.1A Src Range: Best Fixed	Current Range: Limited Auto 1µA Status: Disabled	Speed: Normal Mode: Sampling Interval: 0.1 s #Samples: 60 Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
SMU2: Forcing Function: Common Level: 0 V Compliance: 0.105 A		

Figure 15-68
Definition tab (iv-cap ITM)

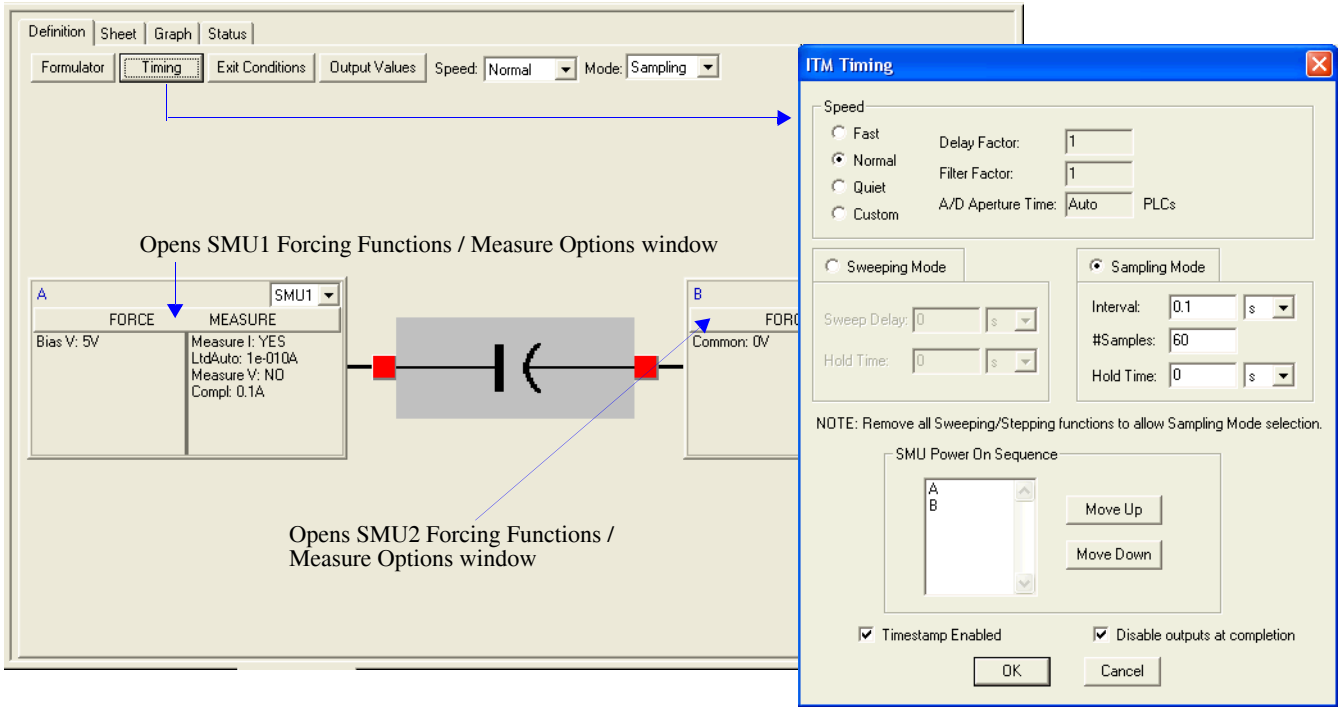
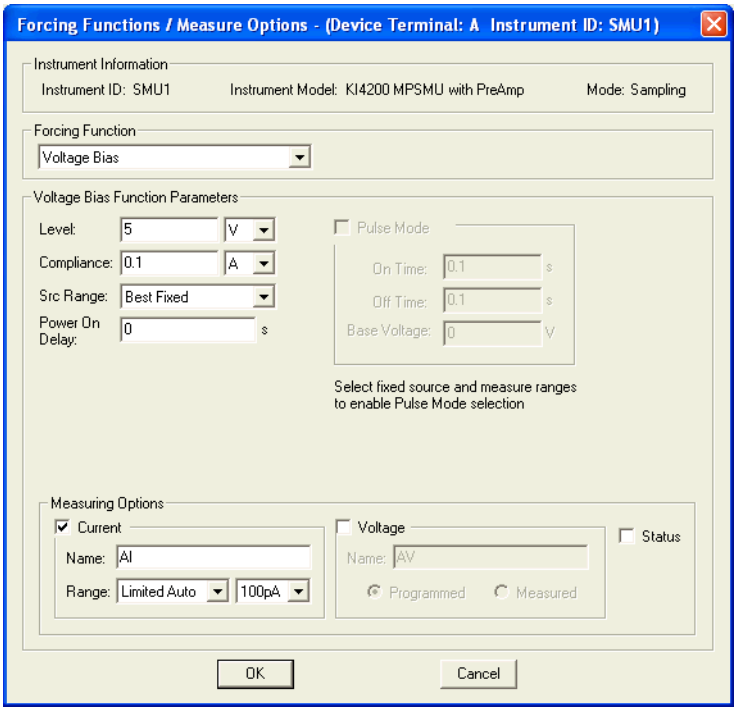


Figure 15-69
Forcing Functions / Measure Options window (iv-cap ITM, SMU1)



Data sheet

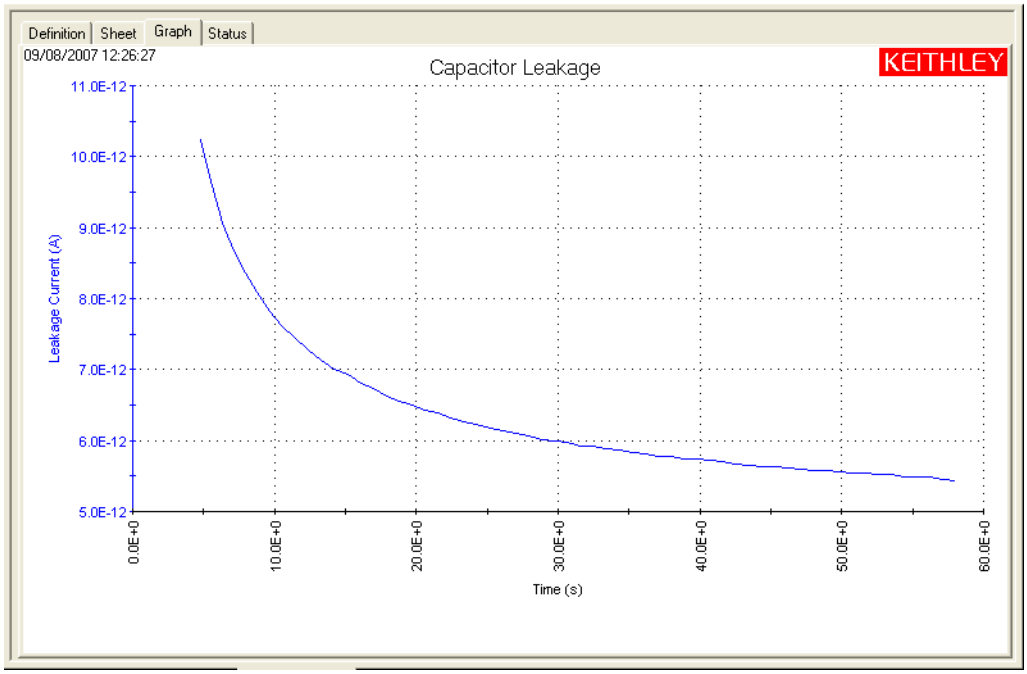
Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Time Timestamp for measurement.
- AI Measured current.

Graph

The graph is displayed in the Graph tab (see Figure 15-70).

Figure 15-70
Graph tab (iv-cap ITM)



cv-cap ITM

Test summary

This ITM applies a DC bias voltage to a capacitor and measures capacitance as a function of time. It generates a Capacitance versus Time graph, and calculates average capacitance and standard deviation.

Parameter settings

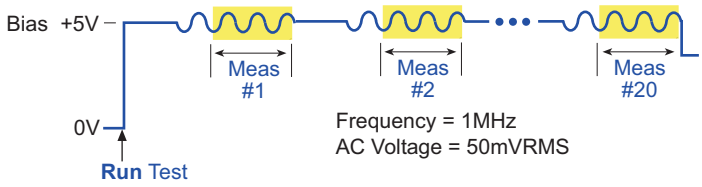
The default parameter settings for this test are listed in Table 15-15. The voltage bias waveform used for this test is shown in Figure 15-71.

Table 15-15
Parameter settings (cv-cap ITM)

Force (Figure 15-73)	**Measure** (Figure 15-73)	**Timing** (Figure 15-72)
CVU Voltage Bias: DC Bias Conditions PreSoak: 0 V DC Bias: 5 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 50 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sampling Interval: 0.1 s #Samples: 20 Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-73)	**CVU Compensation** [3] (Figure 15-73)	
Terminal A: DC Source V Terminal B: AC Measure I (Auto)	Enable Open and Short for this test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-71
Voltage bias waveform (cv-cap ITM)



In the Project Navigator, double-click the **cv-cap** ITM to display the Definition tab, which is shown in Figure 15-72.

Figure 15-72
Definition tab (cv-cap ITM)

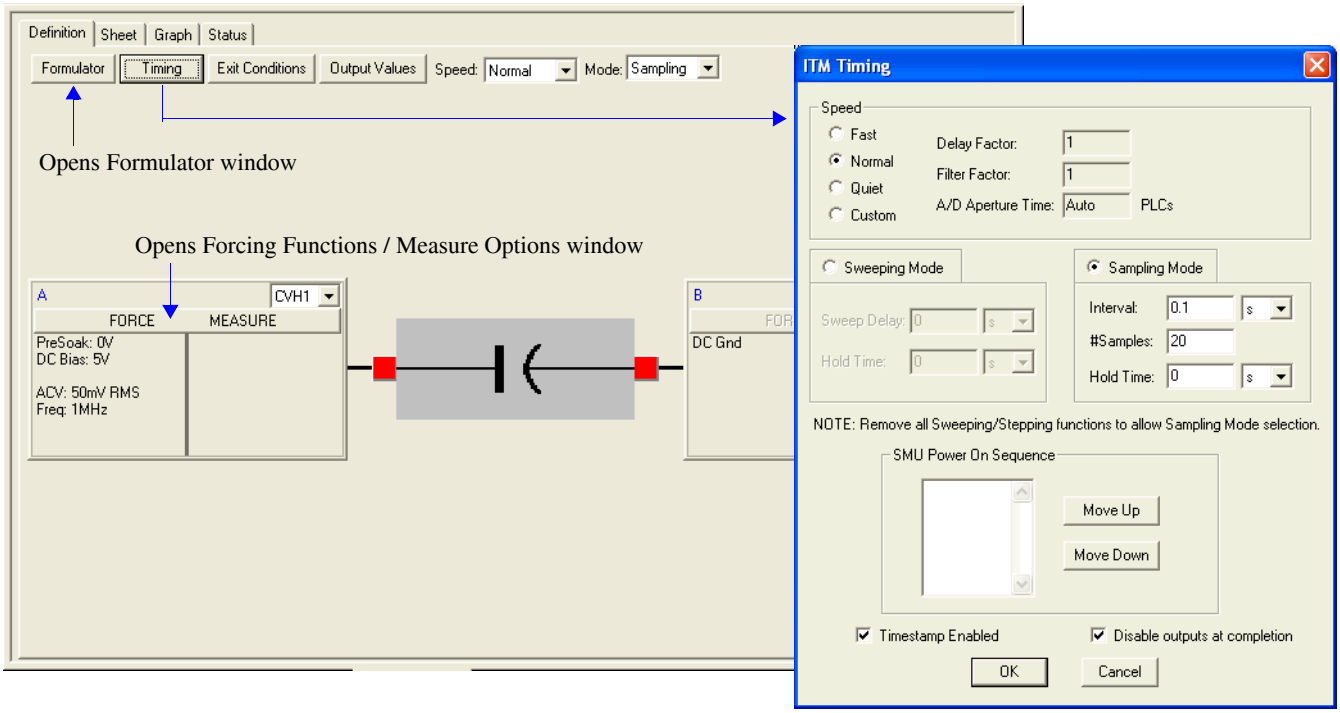
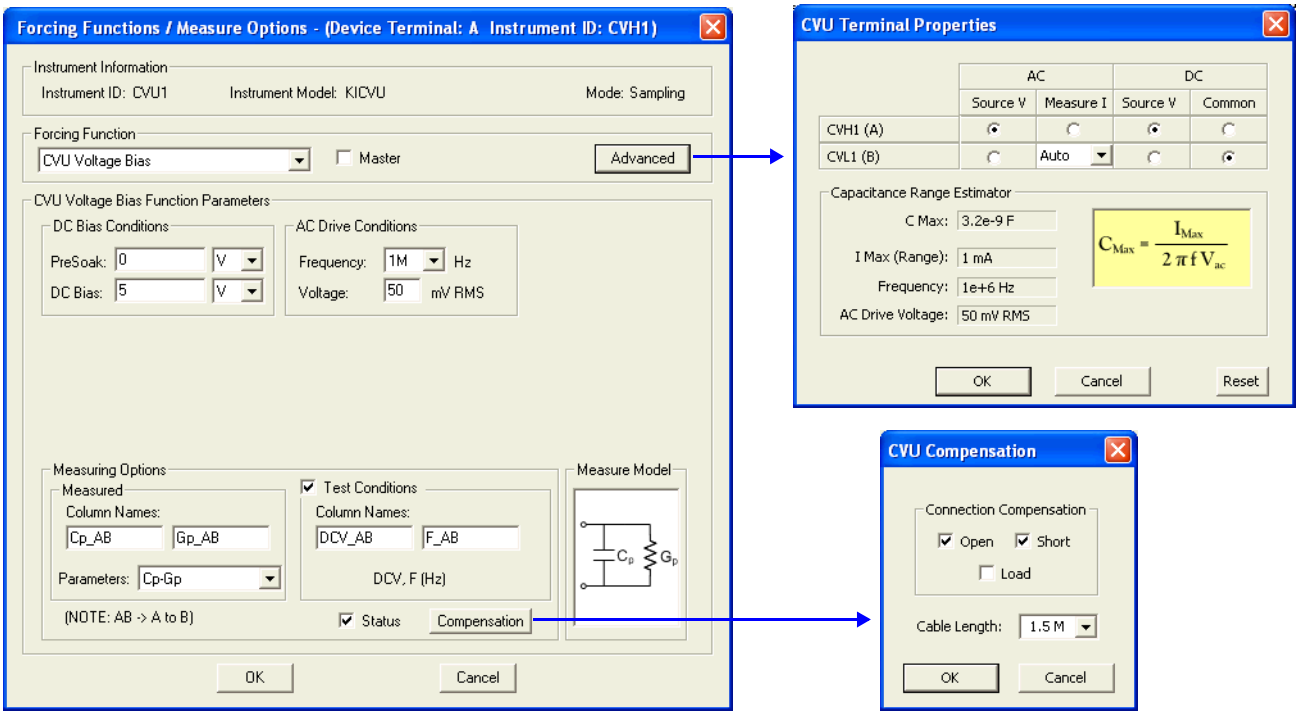


Figure 15-73
Forcing Functions / Measure Options window (cv-cap ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

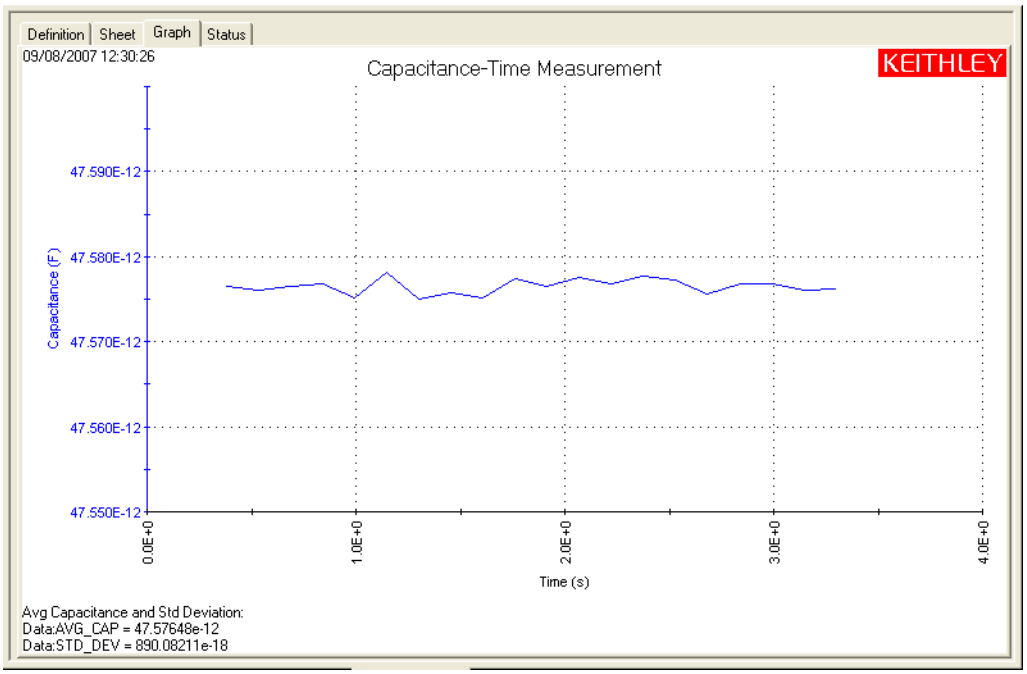
- Time Timestamp for each measurement.
- Cp_AB Measured parallel capacitance.
- Gp_AB Measured conductance.
- DCV_AB Forced DC bias voltage.
- F_AB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- AVG_CAP Formulator calculation result
- STD_DEV Formulator calculation result

Note: AB = Terminal A to Terminal B.

Graph

The graph is displayed in the Graph tab (see Figure 15-74). It includes the formulator results for average capacitance and standard deviation.

Figure 15-74
Graph tab (cv-cap ITM)



CVU_lifetime

Project Plan

Key concepts for generation velocity and lifetime testing (Zerbst plot)

Generation lifetime is an important parameter of metal oxide semiconductor (MOS) capacitors because it determines the storage time of the device. The Zerbst method is a common technique for lifetime measurements. Generating a Zerbst plot requires both a C-V sweep and a C-t sweep.

First the C-V sweep is performed to derive the oxide capacitance (C_{OX}), the minimum capacitance (C_{MIN}), and the doping concentration (N_{AVG} and N_{BULK}). Next the C-t sweep is generated. During the C-t sweep, the MOS capacitor is initially held in accumulation and then biased into deep depletion. While holding this bias, the capacitance is measured as a function of time. From the sweep data, the generation rate is calculated and plotted as a function of the calculated depletion depth.

Creating a Zerbst plot using the Model 4200-CVU

First a C-V sweep is performed and then a capacitance-time (C-t) sweep is performed. The results of the derived parameters (C_{OX}, C_{MIN}, N_{AVG}) from the C-V sweep are integrated with the data taken during the C-t measurements to compute the generation rate and depletion depth.

Project plan tests

This project consists of three ITMs: cv, c-t, and GNI_W-WF.

NOTE *When configuring the project plan, you must input the area of the gate (in units of cm^2) and the dope type (-1 for P-type, 1 for N-type) of the device into the Constants area of the formulator.*

c-v ITM

This test performs a C-V sweep on a MOS capacitor and derives C_{OX}, C_{MIN}, and N_{AVG}. The formulator calculates all the same parameters that are derived in the project plan (CVU_MOScap) to test MOS capacitors.

c-t ITM

This test biases the device in accumulation for a specified period of time, and then the polarity of the voltage source is reversed to drive the device into depletion. The capacitance is plotted as a function of time.

When configuring the test, you will need to set the DC bias hold voltage (for accumulation) and the DC bias voltage level (for depletion). You will also set the hold time for the bias hold voltage (in accumulation).

GNI_W-WF ITM

This test creates a Zerbst plot by first biasing the device in accumulation for a specified time period. At time = 0, the polarity of the bias voltage is reversed to drive the device into deep depletion. While holding this bias, the capacitance is measured as a function of time. As more minority carriers are generated, the measured capacitance will rise. Eventually it will reach the minimum capacitance on the standard C-V curve. From both the C-t and the C-V data, the generation rate (G/n_i) is plotted as a function of depletion depth (w-w_F). You must input the values of C_{OX}, C_{MIN}, and N_{AVG} taken from the cv test into the formulator in order to calculate the generation rate and depletion depth. Note that a known value of N_{AVG} can be input into the formulator instead.

Once G/n_i versus w-w_F is plotted, a linear line fit is applied to the graph. The generation lifetime (τ_G) is the reciprocal of the slope of the linear fitted region of the graph. The surface generation velocity (s_G) is the y-axis (G/n_i) intercept of the same linear section of the Zerbst plot.

The generation rate is calculated as follows:

$$G/ni = -\frac{\varepsilon_S A N_{AVG} C_{OX}}{2}\left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_i t_{int}}\right] \quad \textit{Eq. 1}$$

Where:
- G/ni = Generation rate (s^{-1})
- ε_S = Permittivity of the substrate material (F/cm)
- A = Gate area (cm^2)
- N_{AVG} = Average doping concentration (cm^{-3})*
- C_{OX} = Oxide capacitance (F)
- $C^2_{t(i+1)}$ = (i+1) value of measured C-t capacitance (F)
- $C^2_{t(i-1)}$ = (i-1) value of measured C-t capacitance (F)
- n_i = Intrinsic carrier concentration (cm^{-3})
- t_{int} = Time interval between C-t measurements (s)

* N_{AVG} is calculated (with the result placed in the Sheet tab) when the C-V test is run. The value for this parameter must be input into the formulator for the GNI_W-WF test. Note that a known value of N_{AVG} can be input into the formulator instead.

The equilibrium inversion depth (w_F) is calculated as:

$$w_F = \varepsilon_S A\left(\frac{1}{C_{MIN}} - \frac{1}{C_{OX}}\right) \quad \textit{Eq. 2}$$

Where:
- w_F = Equilibrium inversion depth (cm)
- ε_S = Permittivity of the substrate material (F/cm)
- A = Gate area (cm^2)
- C_{MIN} = Minimum oxide capacitance (F)*
- C_{OX} = Oxide capacitance (F)*

* C_{MIN} and C_{OX} are calculated (with the results placed in the Sheet tab) when the cv test is run. The values for these parameters must be input into the formulator for the GNI_W-WF test.

The depletion depth is calculated from the following equation:

$$w - w_F = \varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}}\right) - w_F \quad \textit{Eq. 3}$$

Where:
- w = Depletion depth (cm)
- w_F = Equilibrium inversion depth (cm)
- ε_S = Permittivity of the substrate material (F/cm)
- A = Gate area (cm^2)
- C_{ti} = i(th) value of measured C-t capacitance (F)
- C_{OX} = Oxide capacitance (F)*

* C_{OX} is calculated (with the result placed in the Sheet tab) when the C-V test is run. The value for this parameter must be input into the formulator for the GNI_W-WF test.

One of the perplexing parts of the Zerbst method is to determine how to space the capacitance measurements. Capacitance measurement spacing almost entirely depends on how fast minority

carriers can be generated. The interval time for the measurements can be adjusted depending on the device.

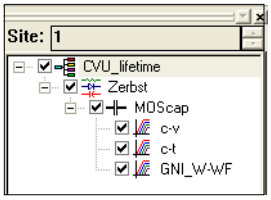
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_lifetime** folder.
5. Open the **CVU_lifetime.kpr** project.

The CVU_lifetime project plan is shown in Figure 15-75.

Figure 15-75
CVU_lifetime project plan

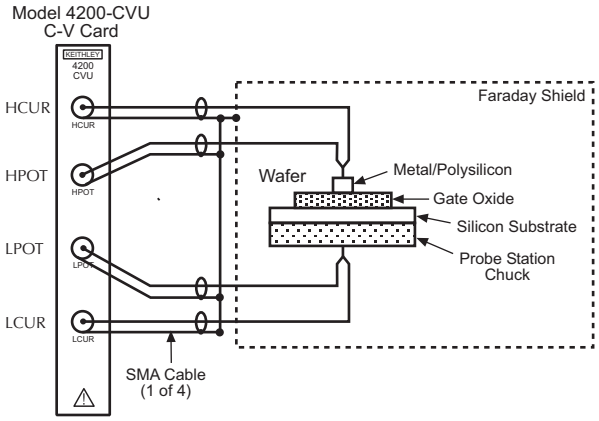


Connections

Figure 15-76 shows the basic test configuration; Connections provides connection details. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-76
Basic configuration for lifetime testing



Formulas and constants

Tests in this project use the same formulas as the ones for the CVU_MOScap project. These formulas are summarized in Table 15-23. Additional formulas (used by the GNI_W-WF test) are summarized in Table 15-16. User-defined constants are summarized in Table 15-17. The formulas and constants are set from the formulators for the tests.

Note that results of the COX, CMIN, and NAVG formulas (used by the cv test) are needed for the GNI_W-WW test, and must be input into the formulator for the GNI_W-WF test by the user.

Table 15-16
Formulas for the GNI_W-WW ITM (CVU_lifetime project)

Formula Name	Formula Details	
GNI (*G/ni*)	Units: s^{-1}	Description: Generation rate
	GNI = - (ES*AREA*NAVG*COX)*DIFF(MAVG(1/CP_GB^2, 5), TIME)/NI	
	Simplified Equation: $G/ni = -\frac{\varepsilon_S A N_{AVG} C_{OX}}{2}\left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_i t_{int}}\right]$ See Eq. 1 for details*	
WF (W_F)	Units: cm	Description: Equilibrium inversion depth
	WF = ES*AREA*(1/MIN(CP_GB)-1/COX)	
	Simplified Equation: $w_F = \varepsilon_S A\left(\frac{1}{C_{MIN}} - \frac{1}{C_{OX}}\right)$ See Eq. 2 for details*	
WWF (w-w_F)	Units: cm	Description: depletion depth (w-w_F computation)
	WWF = ES*AREA*(1/CP_GB-1/COX)-WF	
	Simplified Equation: $w - w_F = \varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}}\right) - w_F$ See Eq. 3 for details*	

* Details for referenced equations are found in the documentation for Key concepts for generation velocity and lifetime testing (Zerbst plot).

Table 15-17
Constants for the CVU_lifetime project

Constant	Default Value	Units	Description
AREA	10.404 E-03	cm^2	Gate area of device
DOPETYPE	1 E+00	none	1 = P-type, -1 = N-type
EBG	1.12 e+00	eV	E_{BG} - Semiconductor energy gap
EOX	340.0 E-15	F / cm	ε_{OX} - Permittivity of oxide
ES	1.04 E-12	F / cm	ε_S - Semiconductor permittivity
NI	14.5 E+09	cm^{-3}	N_I - Intrinsic carrier concentration
TEMP	293 E+00	K	Test temperature
WM	4.15 E+00	V	W_M - Metal work function
WS	4.05 E+00	V	W_S - Silicon electron affinity

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

c-v ITM

Test summary

This ITM performs a C-V sweep on a MOS capacitor and derives C_{OX}, C_{MIN} and N_{AVG} (which are required for the GNI_W-WF test). For more information on this test, see c-v ITM.

Parameter settings

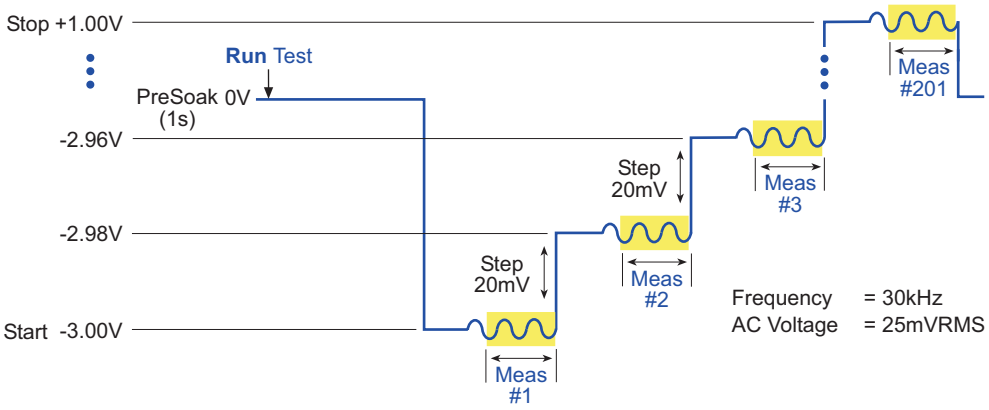
The default parameter settings for this test are listed in Table 15-18. The voltage sweep used for this test is shown in Figure 15-77.

Table 15-18
Parameter settings (c-v ITM)

Force (Figure 15-79)	**Measure** (Figure 15-79)	**Timing** (Figure 15-78)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -3 V Stop: 1 V Step: 0.02 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 25 mV Data Points: 201	Cp-Gp DCV Frequency I Range: Auto [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 1 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-79)	**CVU Compensation** [3] (Figure 15-79)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-77
Linear voltage sweep (c-v ITM)



In the Project Navigator, double-click the **c-v** ITM to display the Definition tab, which is shown in Figure 15-78.

Figure 15-78
Definition tab (c-v ITM)

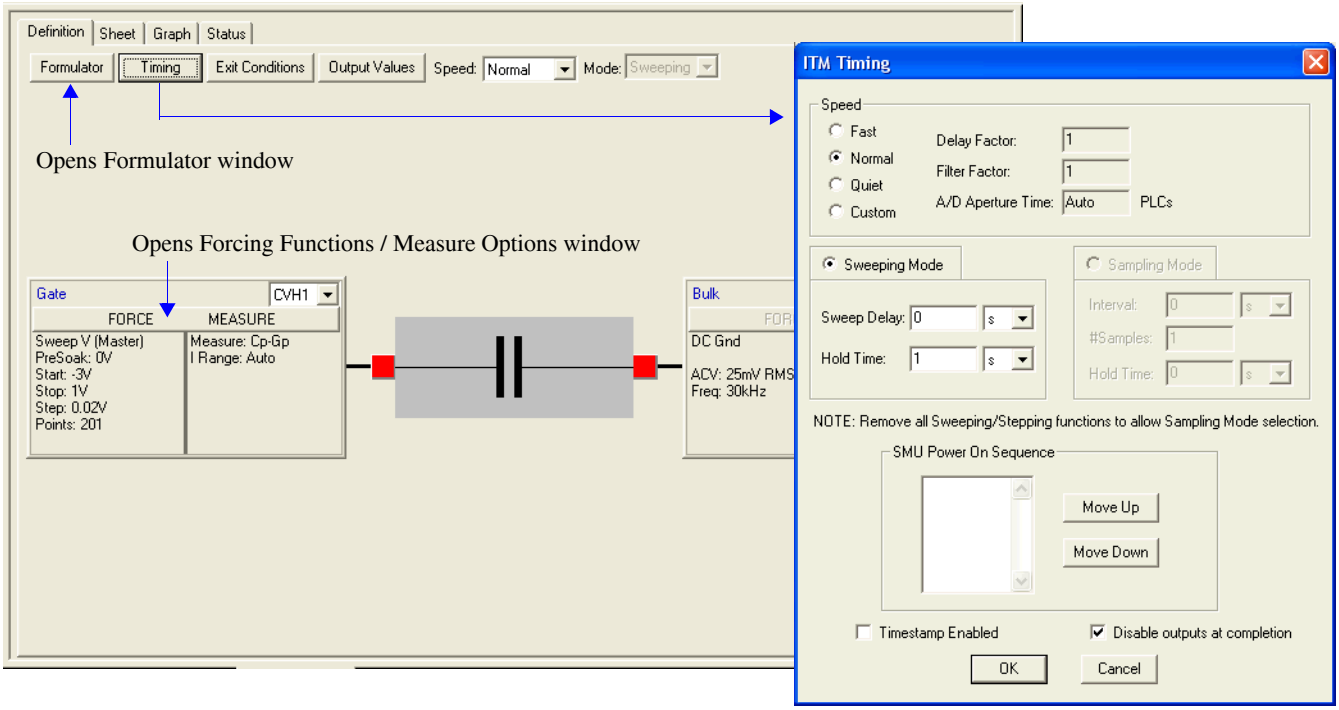
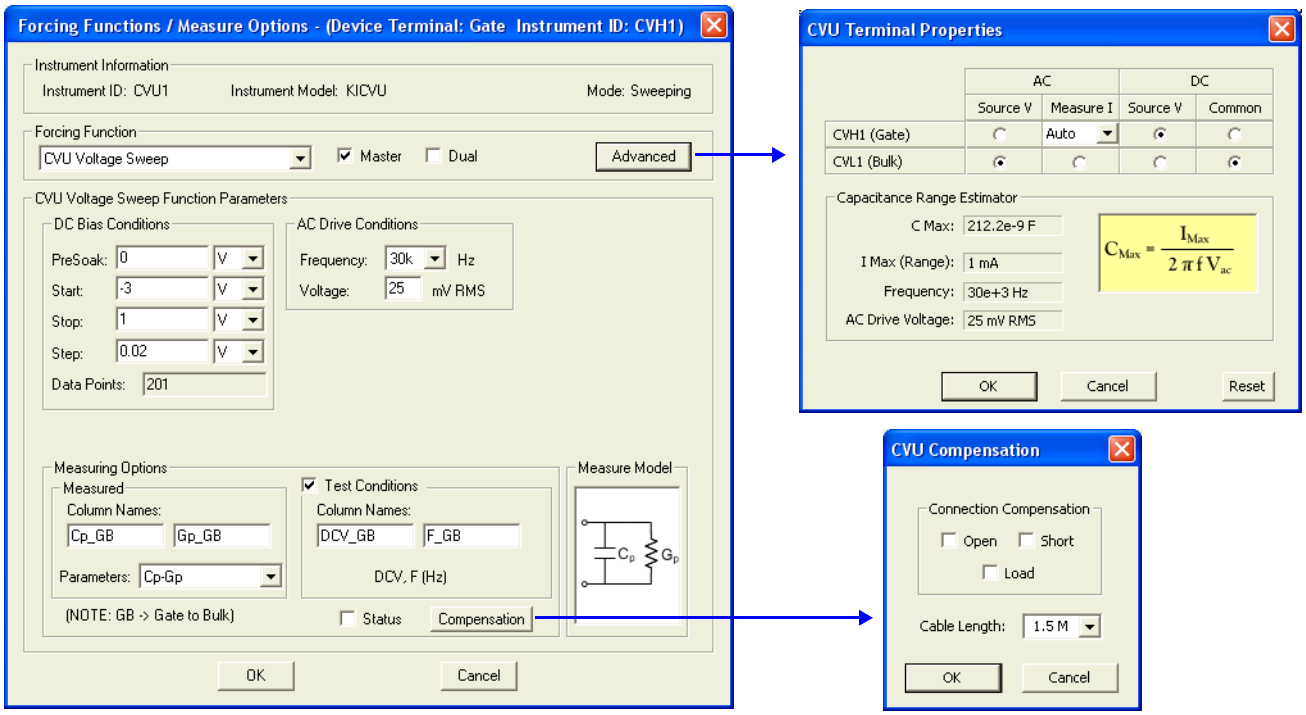


Figure 15-79
Forcing Functions / Measure Options window (c-v ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

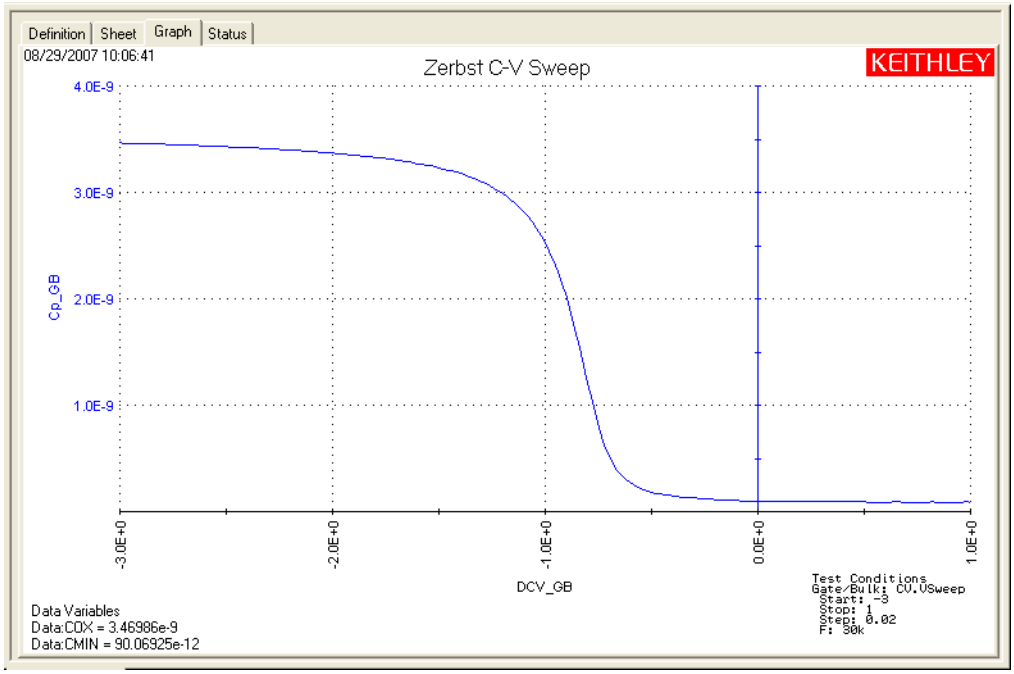
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-80). It includes the calculation results for the COX and CMIN formulas.

Figure 15-80
Graph tab (c-v ITM)



c-t ITM

Test summary

This ITM drives a MOS capacitor into accumulation by applying a negative hold voltage for a period of time (hold time). The bias voltage is then reversed to drive the capacitor into depletion. While in depletion, a series of capacitance measurements are performed at a set time interval. The capacitance versus time measurements are then plotted on a graph. For more information, see c-t ITM.

Parameter settings

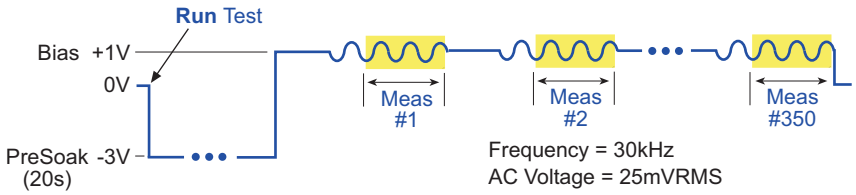
The default parameter settings for this test are listed in Table 15-19. The voltage sweep used for this test is shown in Figure 15-81.

Table 15-19
Parameter settings (c-t ITM)

Force (Figure 15-83)	**Measure** (Figure 15-83)	**Timing** (Figure 15-82)
CVU Voltage Bias: DC Bias Conditions PreSoak: -3 V DC Bias: 1 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 25 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sampling Interval: 1 s #Samples: 350 Hold Time: 20 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-83)	**CVU Compensation** [3] (Figure 15-83)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-81
Voltage bias waveform (c-t ITM)



In the Project Navigator, double-click the **c-t** ITM to display the Definition tab, which is shown in Figure 15-82.

Figure 15-82
Definition tab (c-t ITM)

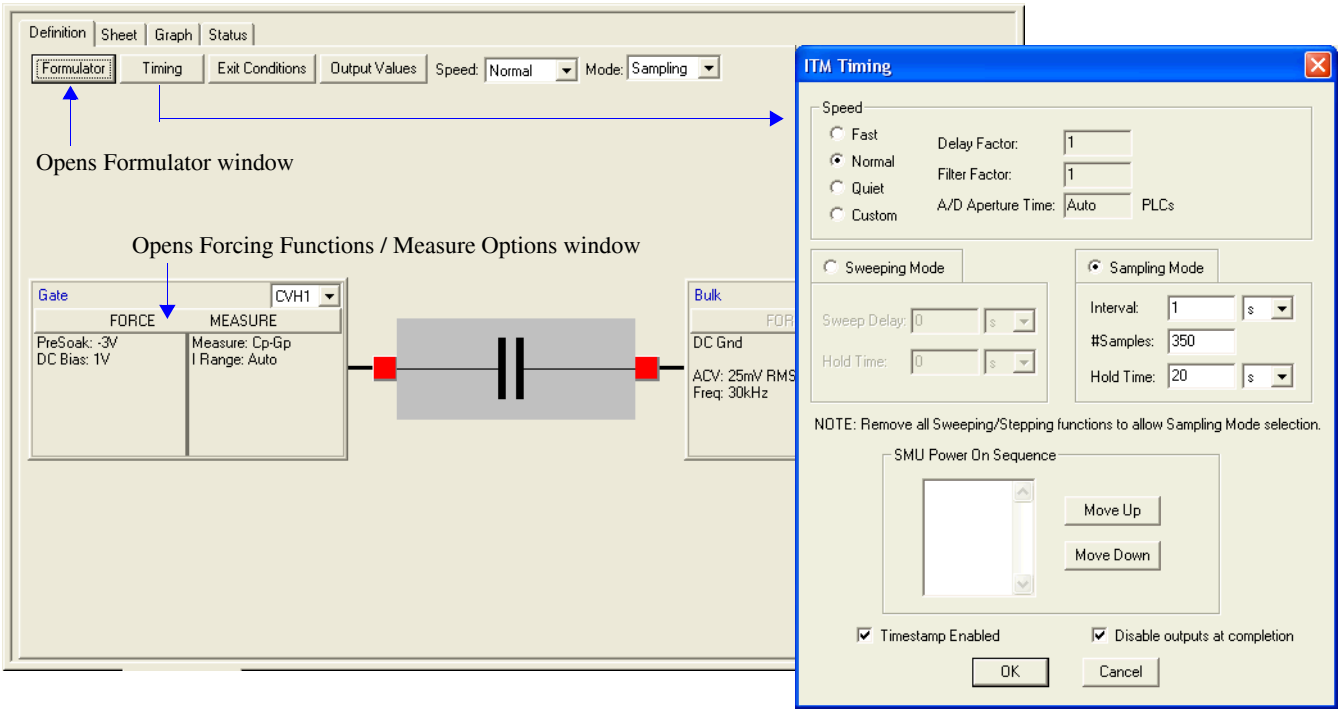
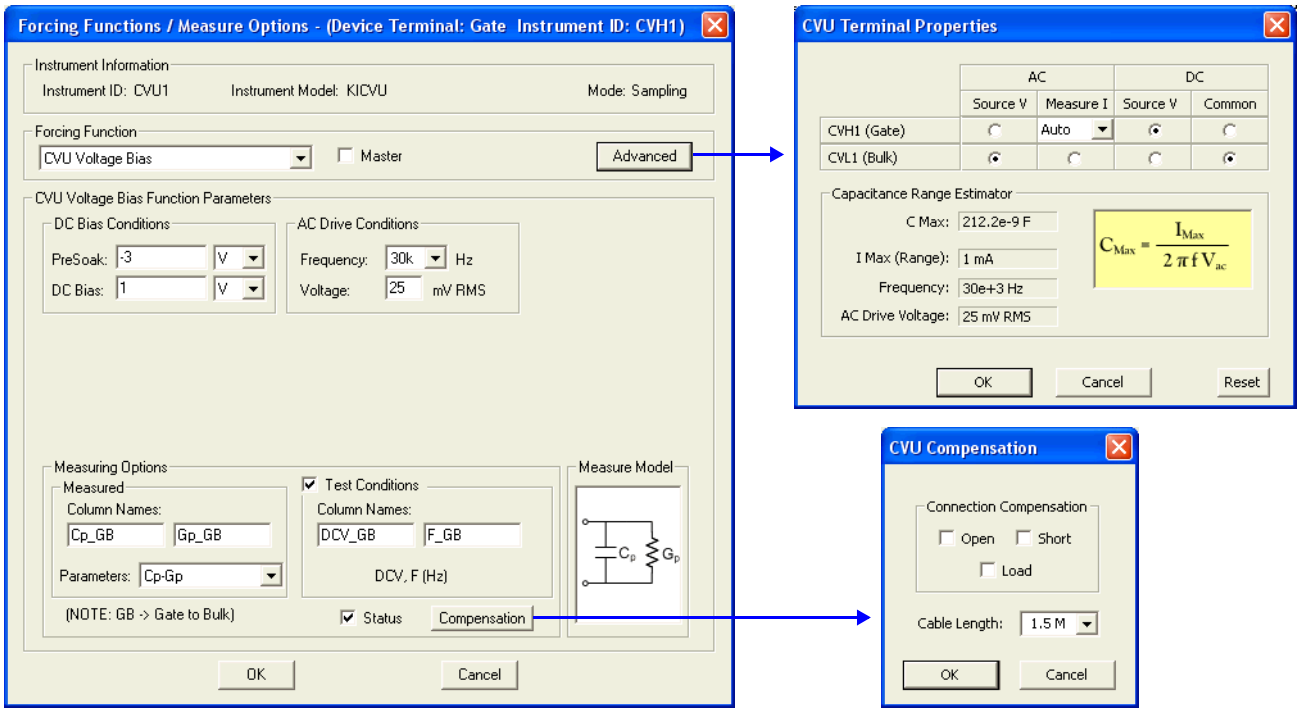


Figure 15-83
Forcing Functions / Measure Options window (c-t ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

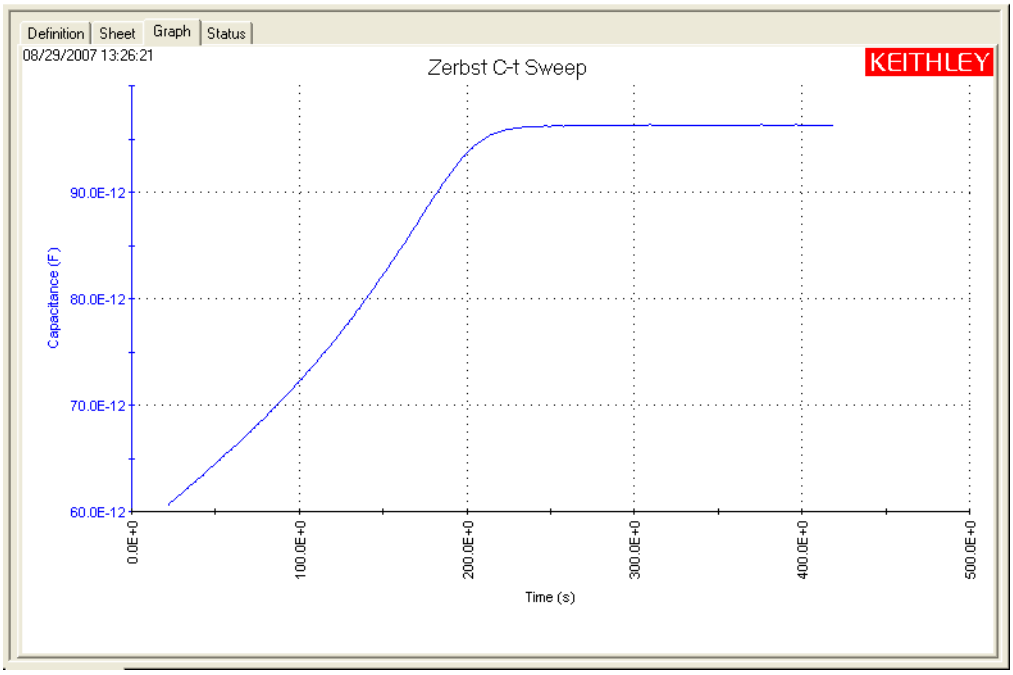
- Time Timestamp for each measurement.
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-84).

Figure 15-84
Graph tab (c-t ITM)



GNI_W-WF ITM

Test summary

This ITM is similar to the c-t test in that it creates a Zerbst plot by first biasing the device in accumulation for a specified time period. However, it calculates the generation rate (GNI) and the depletion depth (WWF) for every measurement, and then generates a GNI vs WWF Zerbst plot.

Procedure

1. Run the C-V ITM.
2. Obtain the calculation results for the COX, CMIN and NAVG formulas from the Sheet tab.
3. Input the values for COX, CMIN and NAVG into the formulator for the GNI_W-WF test (for example, COX = 145E-12).

4. Run the GNI_W-WF test to generate the Zerbst plot.

NOTE *If the value for the NAVG in the formulator is known, it can be used instead of the one derived from the C-V test.*

Parameter settings

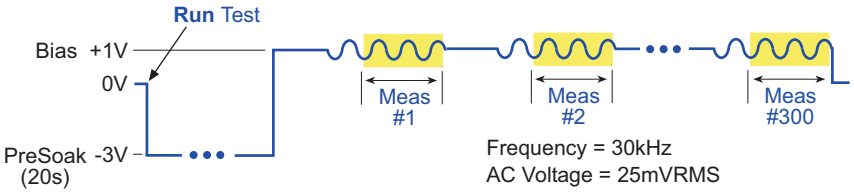
The default parameter settings for this test are listed in Table 15-20. The voltage sweep used for this test is shown in Figure 15-85.

Table 15-20
Parameter settings (GNI_W-WF ITM)

Force (Figure 15-87)	**Measure** (Figure 15-87)	**Timing** (Figure 15-86)
CVU Voltage Bias: DC Bias Conditions PreSoak: -3 V DC Bias: 1 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 25 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sampling Interval: 1 s #Samples: 300 Hold Time: 20 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-87)	**CVU Compensation** [3] (Figure 15-87)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-85
Voltage bias waveform (GNI_W-WF ITM)



In the Project Navigator, double-click the **GNI_W-WF** ITM to display the Definition tab, which is shown in Figure 15-86.

Figure 15-86
Definition tab (GNI_W-WF ITM)

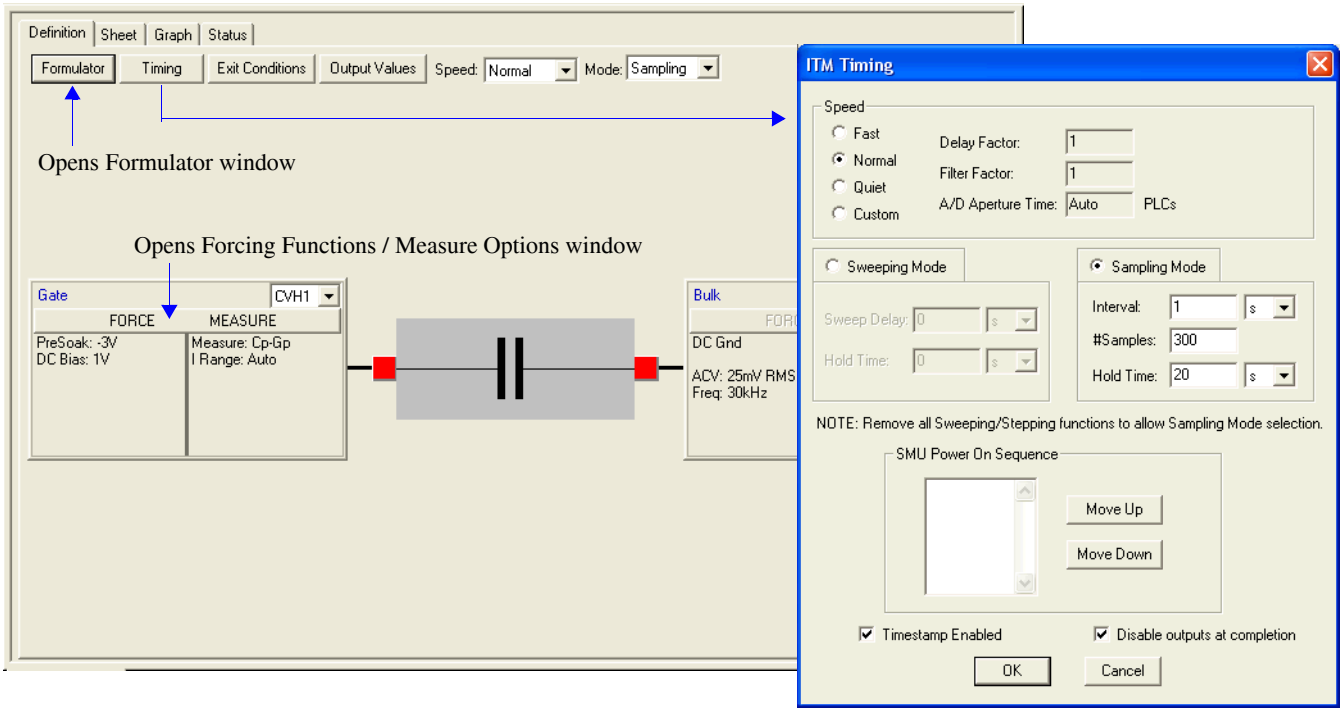
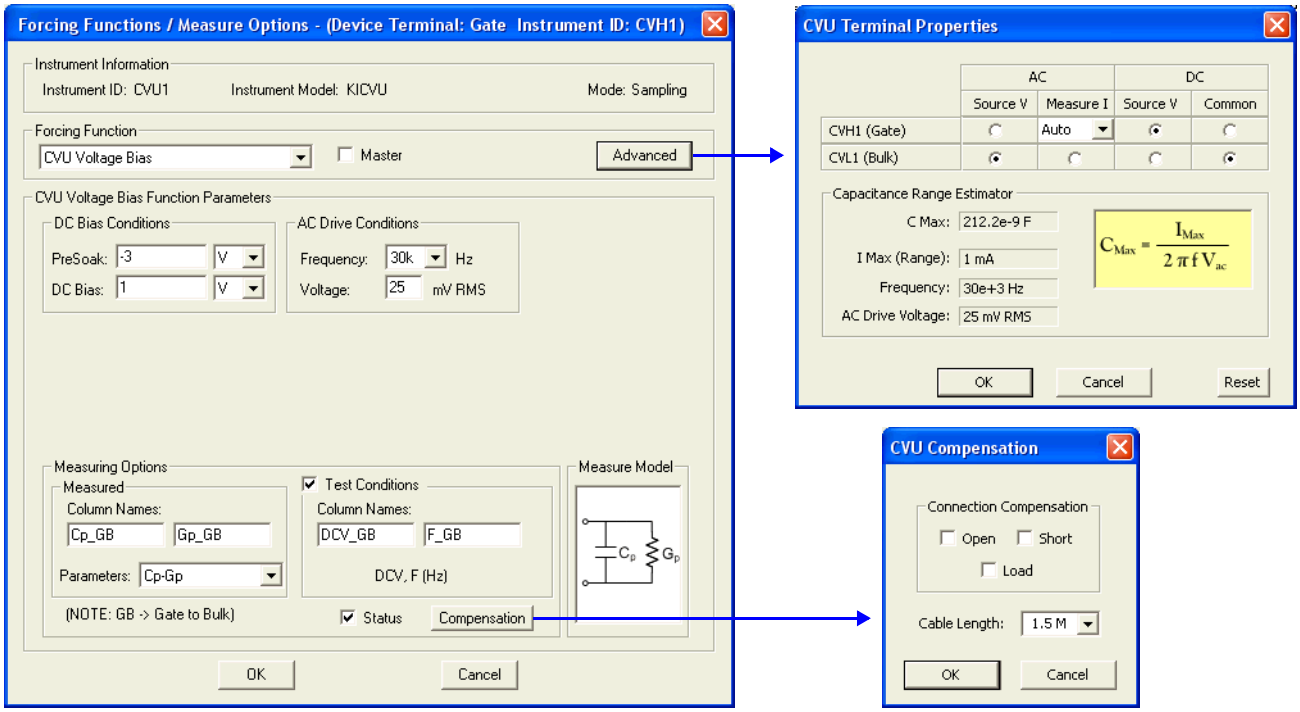


Figure 15-87
Forcing Functions / Measure Options window (GNI_W-WF ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

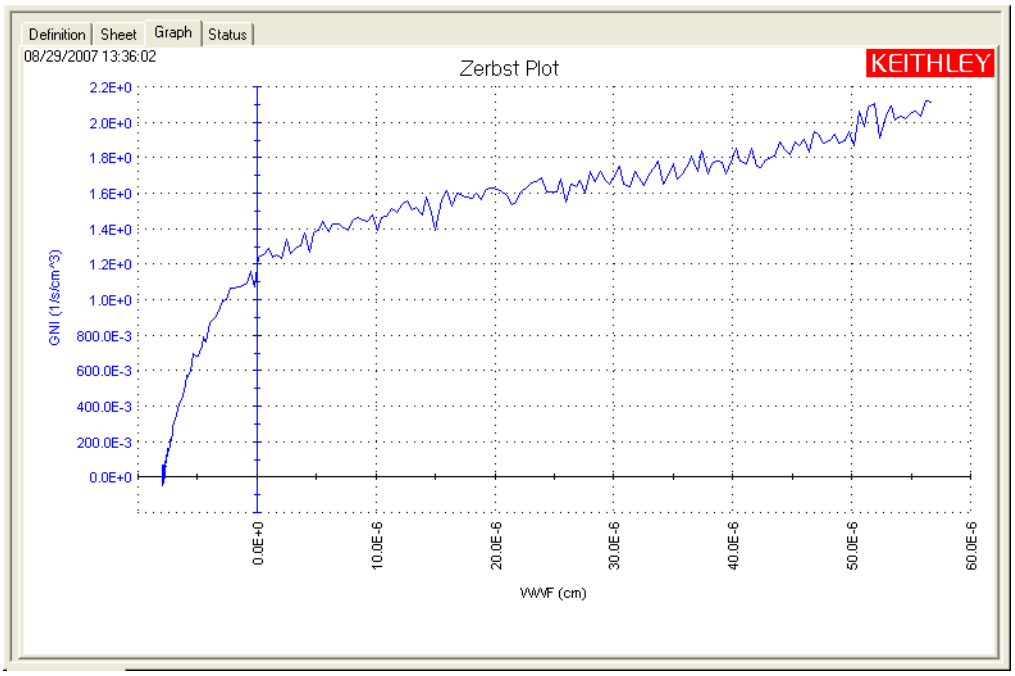
- Time Timestamp for each measurement.
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-88).

Figure 15-88
Graph tab (GNI_W-WF ITM)



CVU_MobileIon

Project Plan

Key concepts for mobile ion measurements (Bias Temperature Stress method)

Mobile ions can present a severe reliability issue in metal oxide semiconductor (MOS) structures. These mobile charges in MOS capacitors are mainly due to ionic impurities, such as Na+, which can drift across the oxide and affect the device performance. One technique for determining the amount of mobile charge in the oxide of a MOS capacitor is the bias temperature stress (BTS) method. Using this method, the flatband voltage (V_{fb}) is used to determine the amount of charge. The flatband voltage is measured both at room temperature and after the device has been at a sufficiently-elevated temperature in order for the charges to be mobile. The difference in the flatband voltage is related to the mobile charge as follows:

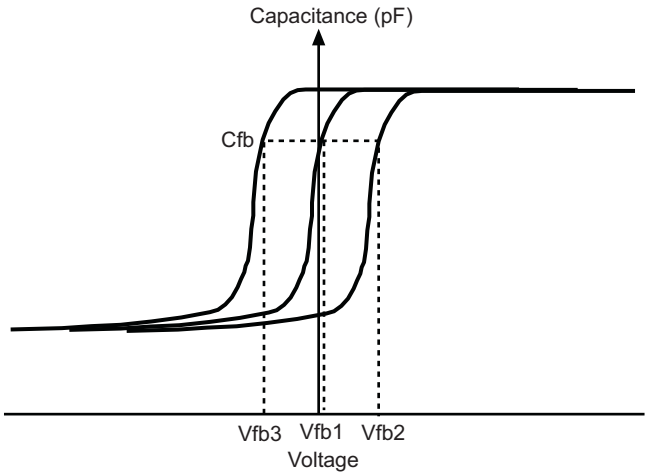
$$Q_m = -\Delta V_{fb} C_{ox}$$

Where: Q_m = Mobile ion charge (C)
V_{fb} = Flatband voltage (V)
C_{ox} = Oxide capacitance (F)

Basic testing involves the following steps:

1. **At room temperature, generate a C-V sweep on the MOS capacitance.** From the C-V data, extract the flatband capacitance (C_{fb}), flatband voltage ($V_{fb}1$), and the oxide capacitance (C_{ox}). The device should be measured on a hot chuck in a dark box. The dark box prevents static noise problems that can affect the C-V measurements.
2. **Apply DC bias voltage and temperature stress.** A gate voltage of approximately 1 MV/cm of dielectric thickness is applied across the device to drive the mobile ions to the interface. A thermal chuck is used to temperature-stress at a temperature in the range of 150°C to 300°C, depending on the type of ions. Typical temperature stress time is around five to 10 minutes. The device is cooled to room temperature with the bias voltage applied. Mobile charges will be trapped near the metal-oxide interface.
3. **Repeat the C-V sweep on a cooled sample and again extract the flatband voltage ($V_{fb}2$).** If there are mobile charges in the oxide, the flatband voltage will be different from the first one.
4. **The sample is heated again, but the opposite polarity DC bias voltage is applied to the sample.** This causes the mobile charges to be driven toward the oxide-silicon interface. Again, the sample is cooled while the DC bias voltage is applied.
5. **Repeat the C-V sweep on the cooled sample and derive the flatband voltage ($V_{fb}3$).** Since mobile charges are concentrated on the opposite interface, this flatband voltage will be different from the first and second measurements. This shift in the flatband voltage is illustrated in Figure 15-89.

Figure 15-89
Flatband voltage shift caused by mobile charges in the oxide



6. The mobile ion charge is calculated as follows:

$$Q_m = \frac{\left|V_{fb}3 - V_{fb}2\right| C_{ox}}{A}$$

Where: Q_m = Mobile ion charge (C)
$V_{fb}2$ = Flatband voltage measured second time after temperature stress (V)
$V_{fb}3$ = Flatband voltage measured third time with opposite bias polarity (V)
C_{ox} = Oxide capacitance (F)
A = Gate area (cm^2)

The mobile ion concentration is related to the mobile ion charge through the following equation:

$$N_{mi} = Q_m / q$$

Where: N_{mi} = Mobile ion concentration
Q_m = Mobile ion charge (C)
Q = Electron charge

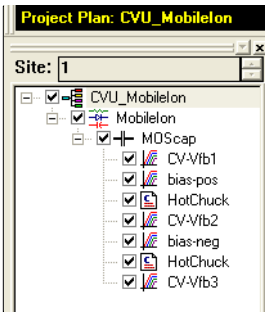
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_MobileIon** folder.
5. Open the **CVU_MobileIon.kpr** project.

The CVU_MobileIon project plan is shown in Figure 15-90.

Figure 15-90
CVU_MobileIon project plan



Project test summaries

This project plan uses the following seven test modules to measure mobile ions:

CV-Vfb1 ITM

This ITM performs a basic C-V measurement on the device and extracts flatband capacitance, flatband voltage, and the oxide capacitance. The oxide capacitance and the flatband voltage are sent to the subsite level spreadsheet, and can be used in the mobile ion calculation. To set these values, open the **Definition tab**, click the **Output Values** button, and select the desired value in the ITM Output Values dialog box (Figure 15-94 shows that COX is enabled as an Output Value).

bias-pos ITM

This ITM applies a positive DC bias voltage to the sample. The voltage continues to be applied to the sample as the device under test (DUT) is heated in the next test module, HotChuck.

HotChuck UTM

This UTM prompts you to turn on the hot chuck to a desired temperature and then to cool down the sample. After the temperature stress, click **OK** to generate the C-V sweep in the following test. The bias voltage continues to output until you click **OK**.

CV-Vfb2 ITM

This ITM performs C-V sweep on sample after it has been heated and then cooled. The flatband voltage is sent to the subsite level data sheet by selecting the appropriate setting in the ITM Output Values dialog box.

bias-neg ITM

This ITM applies a negative polarity DC bias voltage to the sample. The voltage continues to be applied to the sample as the DUT is heated in the next test module, HotChuck.

HotChuck UTM

This UTM prompts you to turn on the hot chuck to a desired temperature. After reaching that temperature, turn off the hot chuck to cool down the sample. When the sample is cooled, click **OK** to generate the C-V sweep in the test that follows. The bias voltage continues to output until you click **OK**.

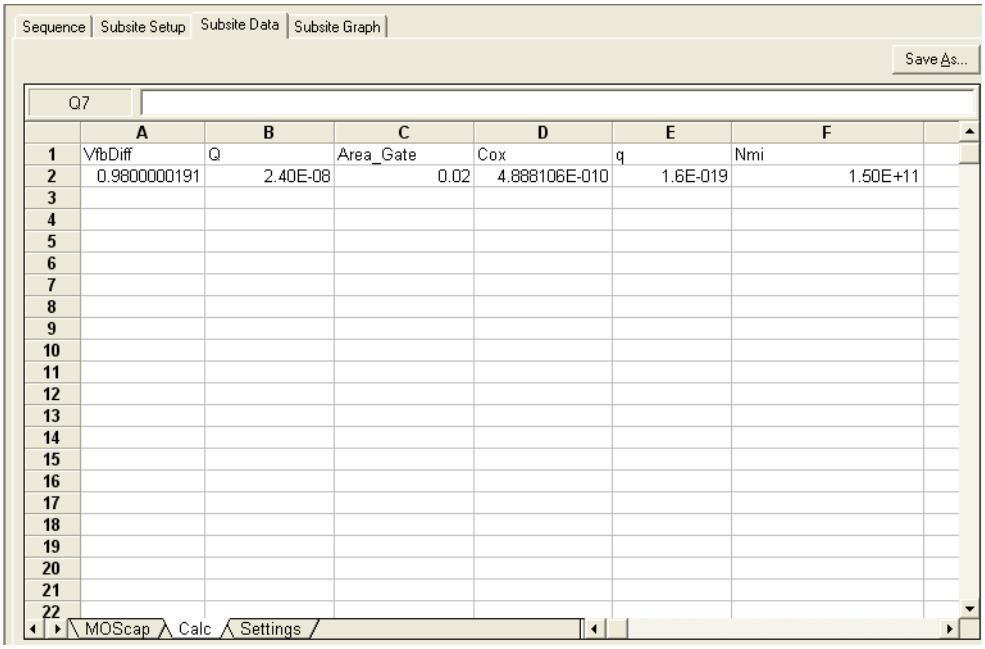
CV-Vfb3 ITM

This ITM generates a third C-V sweep for the cooled sample. The flatband voltage is sent to the subsite calculation sheet by selecting the appropriate setting in the ITM Output Values dialog box.

The calculation sheet for the subsite is opened as follows:

1. Double-click **MobileIon** in the project navigator shown in Figure 15-90.
2. To open the calculation sheet (shown in Figure 15-91), click the **Subsite Data** tab (located at the top of the sheet), and then click the **Calc** tab (located at the bottom of the sheet).

Figure 15-91
Calc sheet for the subsite level (MobileIon)



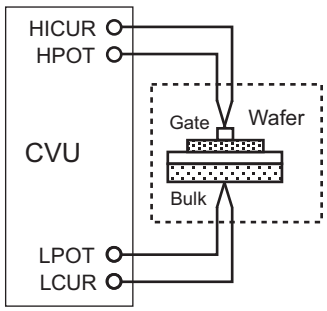
	A	B	C	D	E	F
1	VfbDiff	Q	Area_Gate	Cox	q	Nmi
2	0.9800000191	2.40E-08	0.02	4.888106E-010	1.6E-019	1.50E+11

Connections

Figure 15-92 shows the basic test configuration; Connections provides details on connections to a semiconductor wafer. Only use the supplied (red), same length (1.5 m or 3 m), 100 Ω SMA cables for connections to the Model 4200-CVU.

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-92
Basic configuration for mobile ion testing



Formulas and constants

This project uses formulas and constants that are used by the Model 4200-CVU_MOScap project plan. The formulas and constants are summarized in Table 15-23 and Table 15-24. See Key concepts for MOScap testing for details on formulas used to test MOS capacitors.

The CV-Vfb1 ITM also uses the following formula for gate area:

AREA_GATE = 2

Running the subsite plan

This semi-automatic test is executed from the subsite level of the project plan. The test is run from this level because the oxide thickness and flatband voltage extracted from the data in the ITMs is sent to the subsite data sheet after the seven test modules are executed. The calculated value of the mobile ion charge and concentration can be found in the Calc sheet tab of the Subsite Data sheet (see Figure 15-91).

Before executing this subsite plan, you need to input the gate area of the device into the Constants area of the formulator Window for the CV-Vfb1, CV-Vfb2, and CV-Vfb3 test modules. Also, for the CV-Vfb1 ITM, you must also update the gate area in the formula area of the formulator window. Figure 15-94 shows how to open the formulator. The gate area formula to update for the CV-Vfb1 ITM is named AREA_GATE and is used in the calculation of the mobile ion charge in the subsite Calc sheet.

Running project plan tests provides the basic procedure to run the project plan tests. Keep in mind that this project must be run from the MobileIon subsite level.

NOTE *The documentation for the individual tests for this project plan is provided in the following paragraphs:*

CV-Vfb1, CV-Vfb2, and CV-Vfb3 ITMs

Test summaries

Test summaries for these Interactive Test Modules (ITMs) are provided in Project test summaries (see CV-Vfb1 ITM, CV-Vfb2 ITM, and CV-Vfb3 ITM). In general, these are the tests that perform C-V measurements on the device and extract flatband capacitance, flatband voltage, and oxide capacitance.

Parameter settings

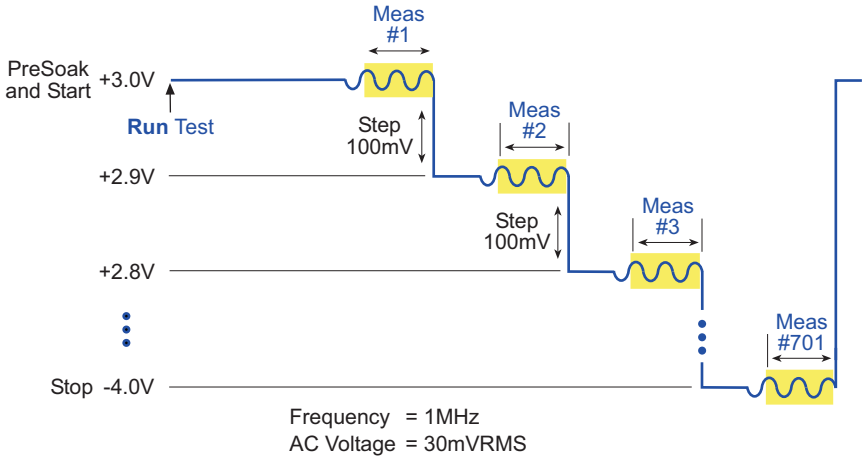
The default parameter settings for these tests are listed in Table 15-9. The voltage sweep used for these tests is shown in Figure 15-93.

Table 15-21
Parameter settings (all three ITMs)

Force (Figure 15-95)	**Measure** (Figure 15-95)	**Timing** (Figure 15-94)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 3 V Start: 3 V Stop: -4 V Step: -01 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 30 mV Data Points: 701	Cp-Gp DCV Frequency I Range: 1 mA [1] Status: Disabled	Speed: Quiet Mode: Sweeping Sweep Delay: 0.1 s Hold Time: 25 s Timestamp: CV-Vfb1: Enabled CV-Vfb2 and CV-Vfb3: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-95)	**CVU Compensation** [3] (Figure 15-95)	
Gate: DC Source V Gate: AC Measure I (1 mA)	Enable compensation as required for the test. Set the cable length being used.	
Output Values [4] (Figure 15-94)		
CV-Vfb1 ITM: COX, VFB, and AREA_GATE CV-Vfb2 and CV-Vfb3 ITMs: VFB		

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.
4. The Output Values are used for passing parameter values to the subsite level Calc sheet. See Section 6 (Keithley Interactive Test Environment) for details on Output Values.

Figure 15-93
Linear voltage sweep (all three ITMs)



In the Project Navigator, double-click the **CV-Vfb1**, **CV-Vfb2**, or **CV-Vfb3** ITM to open the Definition tab shown in Figure 15-94.

Figure 15-94
Definition tab (all three ITMs)

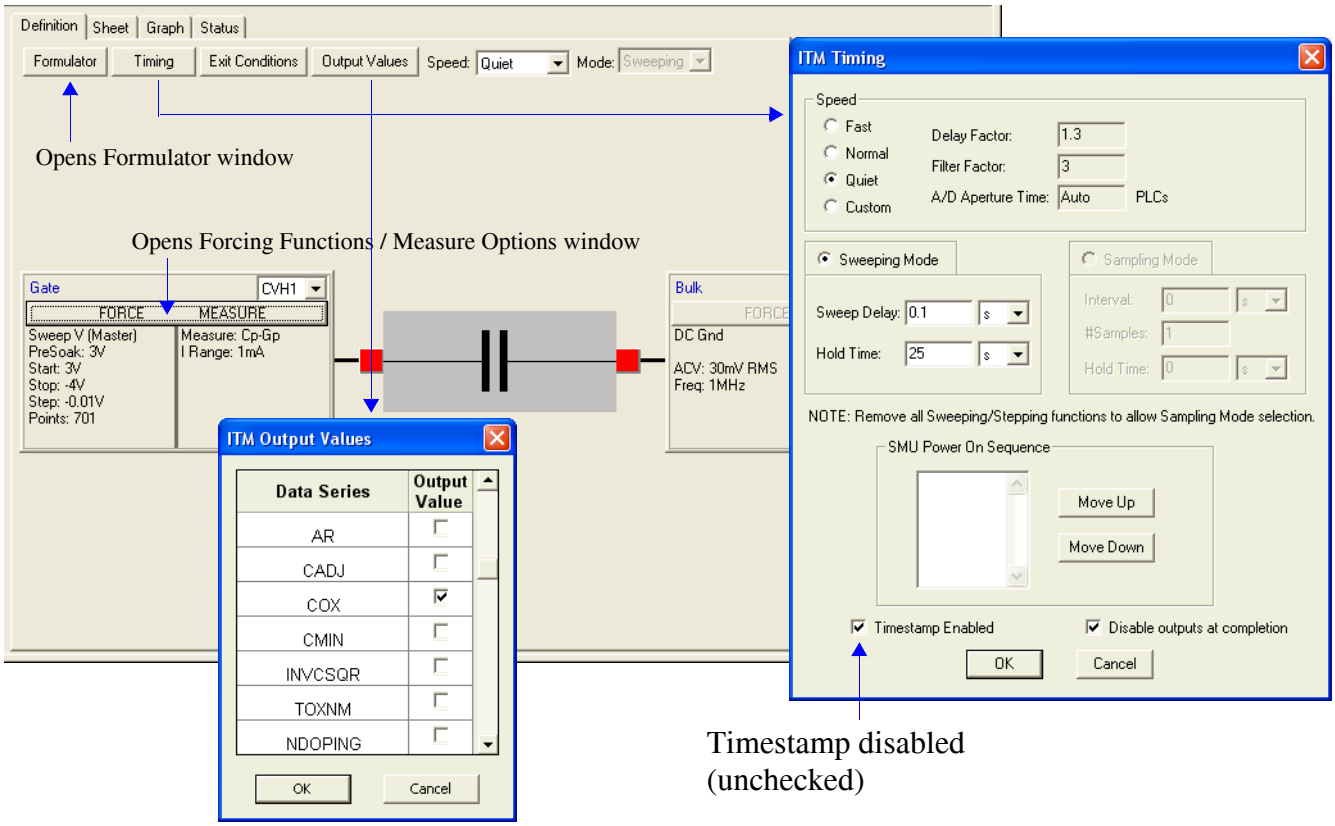
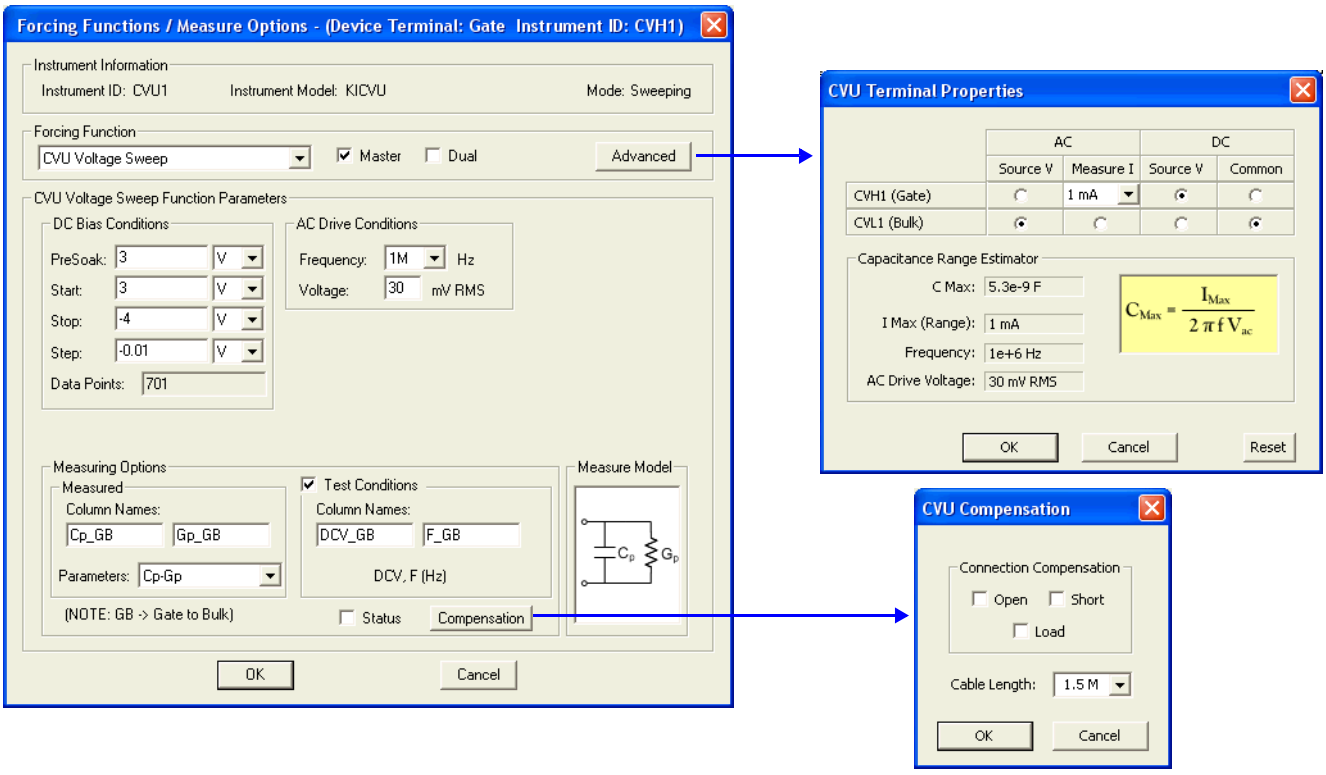


Figure 15-95
Forcing Functions / Measure Options window (all three ITMs)



Formulator formulas and constants

Formulas and constants are defined in the Formulator, which is opened from the Definition tab (see Figure 15-94). See Formulas and constants for details on formulas and constants.

Data sheets

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Time Timestamp for each measurement (CV-Vfb ITM only).
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graphs

The graphical results for the C-V sweep tests are displayed in the Graph tabs:

- Figure 15-96 shows the graph for the CV-Vfb1 test.
- Figure 15-97 shows the graph for the CV-Vfb2 test.
- Figure 15-98 shows the graph for the CV-Vfb3 test.

Figure 15-96
Graph tab (CV-Vfb1 ITM)

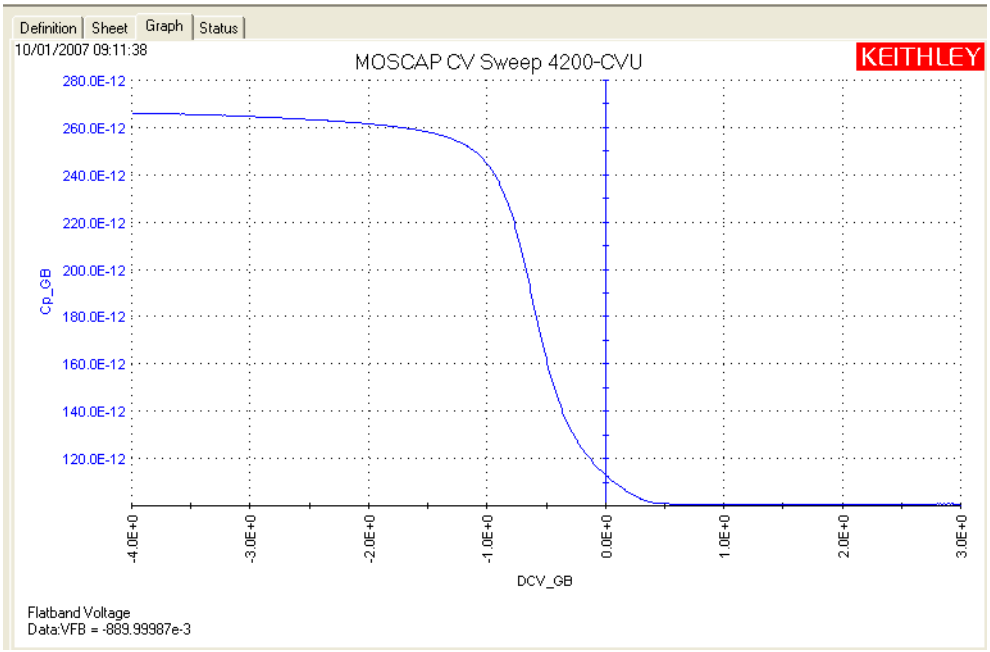


Figure 15-97
Graph tab (CV-Vfb2 ITM)

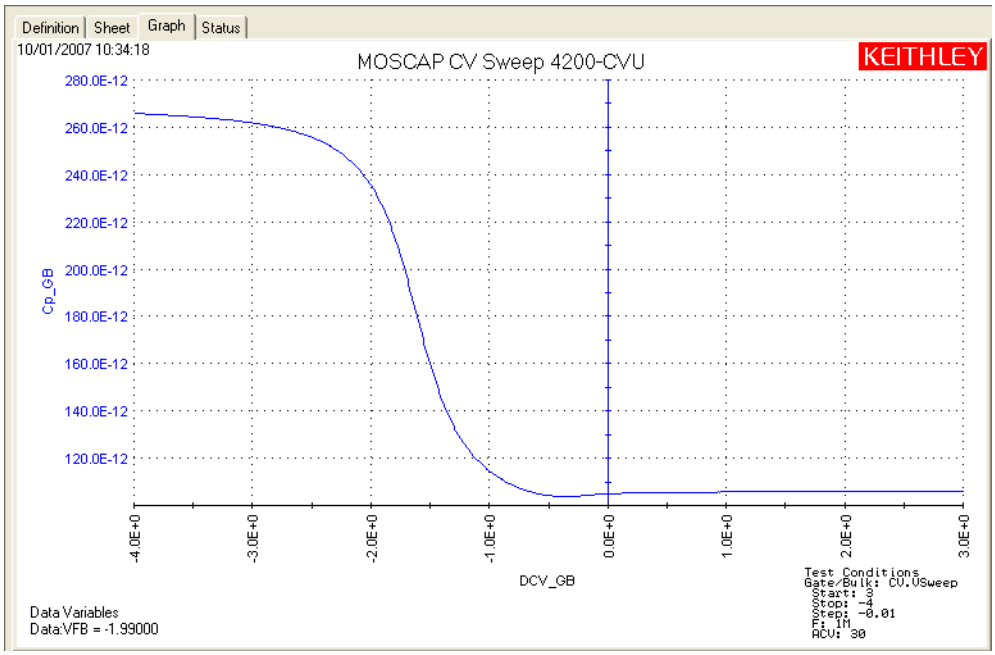
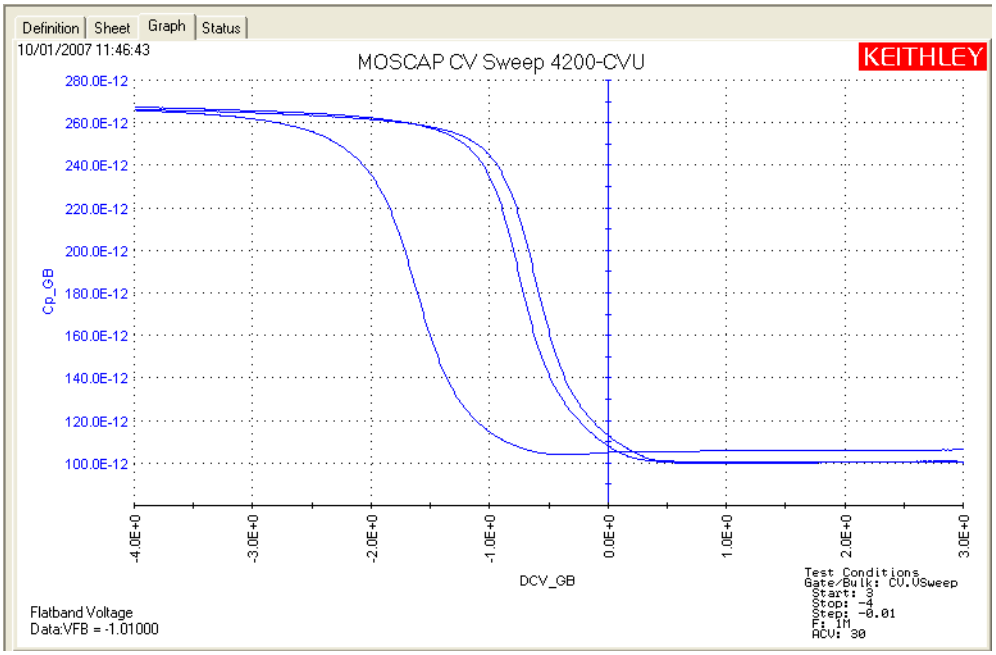


Figure 15-98
Graph tab (CV-Vfb3 ITM)



bias-pos and bias-neg ITMs

Test summaries

Test summaries for these Interactive Test Modules (ITMs) are provided in Project test summaries (see bias-pos ITM and bias-neg ITM). In general, these are the tests that provide the +5 V and -5 V DC bias stress voltages for the test.

Parameter settings

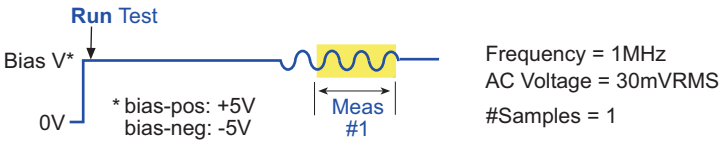
The default parameter settings for these tests are listed in Table 15-22. The DC bias voltage waveform used for these tests is shown in Figure 15-99.

Table 15-22
Parameter settings (both ITMs)

Force (Figure 15-101)	**Measure** (Figure 15-101)	**Timing** (Figure 15-100)
CVU Voltage Bias: DC Bias Conditions PreSoak: 0 V DC Bias: bias-pos: 5 V bias-neg: -5 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 30 mV	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sampling Interval: 0 s #Samples: 1 Hold Time: 10 s Timestamp: Enabled Outputs at completion: Enabled (not checked)
Advanced (Terminal Properties) [2] (Figure 15-101)	**CVU Compensation** [3] (Figure 15-101)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-99
Voltage bias waveform (both ITMs)



In the Project Navigator, double-click the **bias-pos** or **bias-neg** ITM to open the Definition tab shown in Figure 15-100.

Figure 15-100
Definition tab (both ITMs)

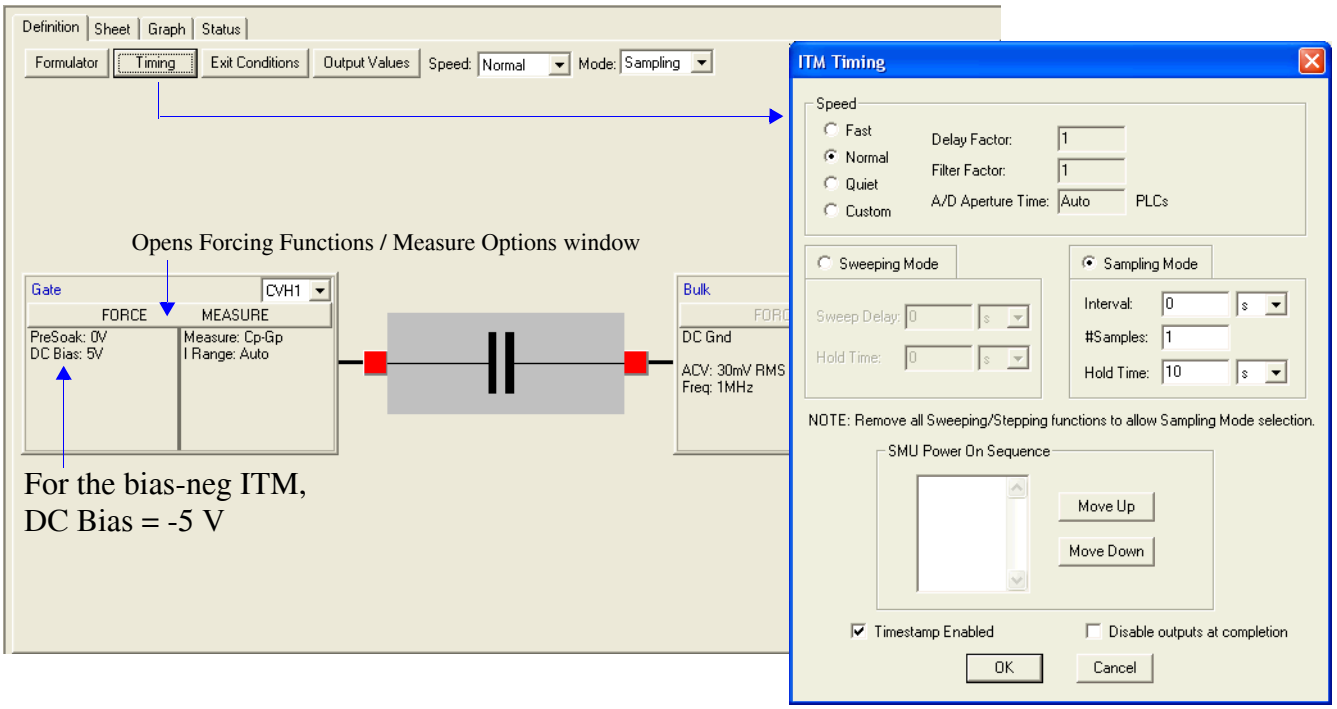
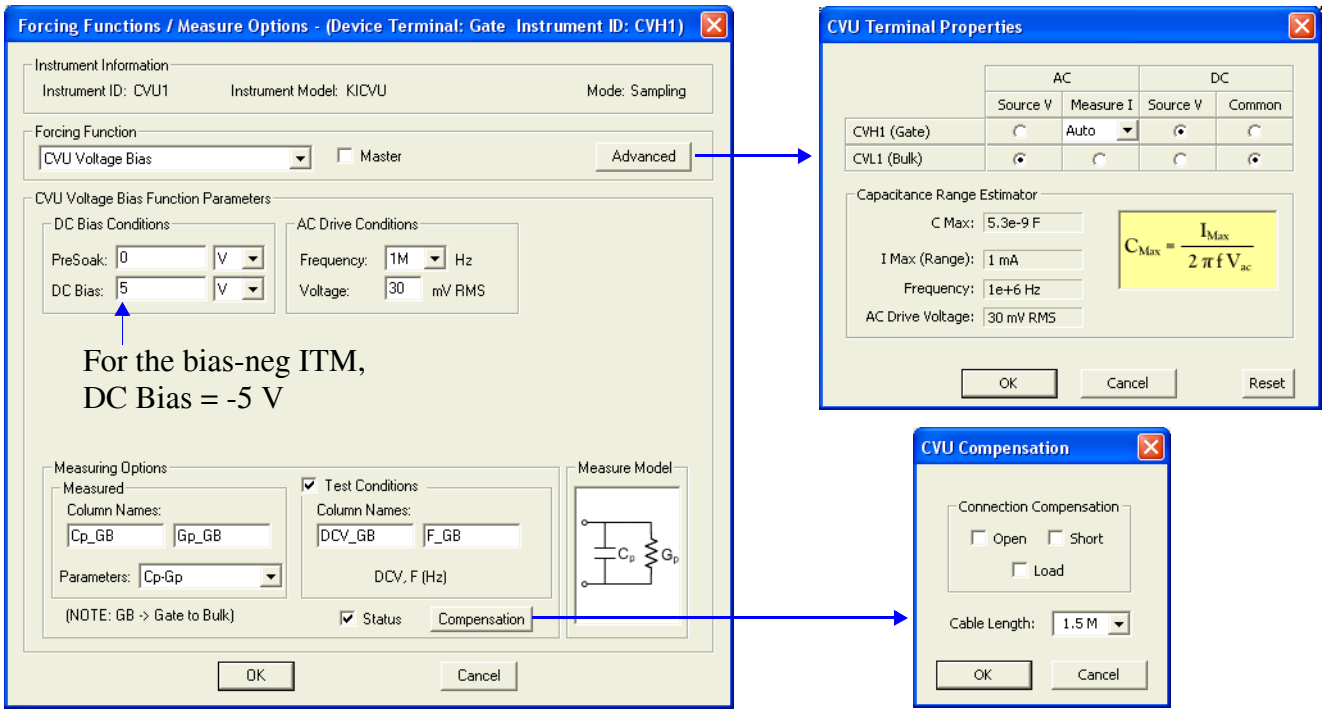


Figure 15-101
Forcing Functions / Measure Options window (both ITMs)



Formulator formulas and constants

Formulas are not used for this test.

Data sheets

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Time Timestamp for the measurement.
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for the measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.

Note: GB = gate-to-bulk.

Graphs

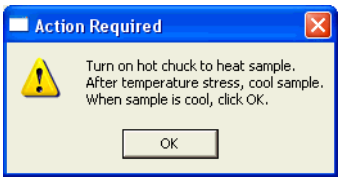
Graphs are not generated for these tests.

HotChuck UTM

Test summary

The test summary for this UTM is provided in the Project test summaries (see HotChuck UTM). In general, these are the prompts used between the second and third bias-sweep tests for temperature control of the chuck and sample. The dialog box for the prompts is shown in Figure 15-102.

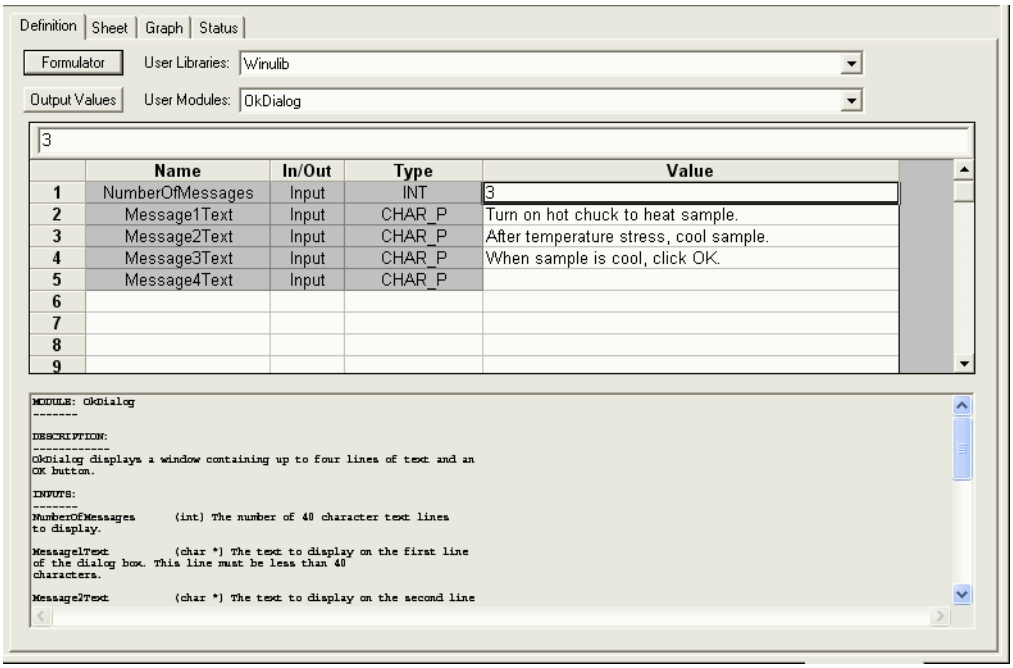
Figure 15-102
Project prompt



Definition tabs

The dialog box prompt is created in the Definition tab of a UTM, which is opened by double-clicking the UTM in the project navigator. Figure 15-103 shows the Definition tab for the HotChuck UTM. Appendix A of this manual provides details on creating project prompts.

Figure 15-103
Definition tab (HotChuck UTM)



CVU_MOScap

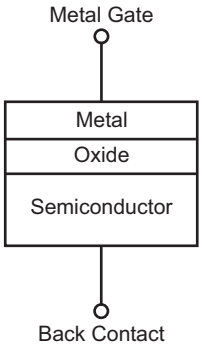
Project Plan

Key concepts for MOScap testing

Maintaining the quality and reliability of gate oxides of MOS structures is a critical task in semiconductor fabrication. A commonly used tool for studying gate-oxide quality in detail is the capacitance-voltage technique. C-V measurements are typically made on a capacitor-like device called a MOS capacitor, or MOS-C.

An example of the construction of a MOS capacitor is shown in Figure 15-104. As shown, the MOS capacitor is just an oxide placed between a semiconductor and a metal gate. The semiconductor and the metal gate are the two plates of the capacitor. The oxide functions as the dielectric. The area of the metal gate defines the area of the capacitor.

Figure 15-104
MOS capacitor

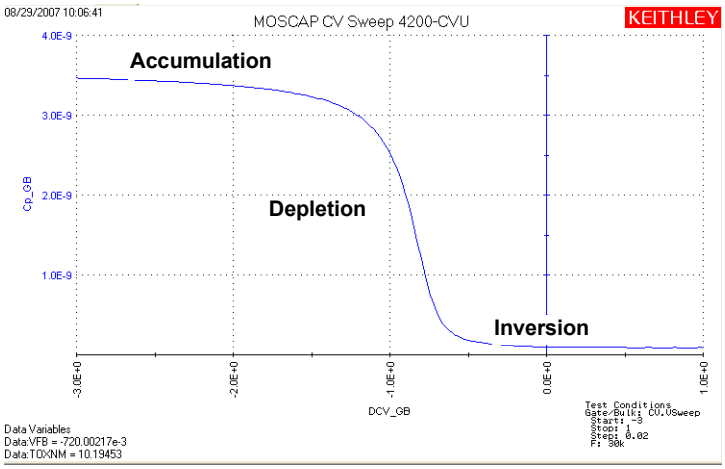


The most important property of the MOS capacitor is that its capacitance changes with changes to applied voltage. As a result, the modes of operation of the MOS capacitor change as a function of the applied voltage. As a DC sweep voltage is applied to the gate, it causes the device to pass through accumulation, depletion, and inversion.

Understanding MOS capacitor C-V curves

Figure 15-105 illustrates a high-frequency C-V curve for a p-type semiconductor substrate. A C-V curve can be divided into three regions: accumulation, depletion, and inversion. Each of the three regions is described for a p-type MOS-C.

Figure 15-105
C-V curve example for p-type MOS-C



The C-V curve for an n-type MOS-C is analogous to a p-type, except that 1) the majority carriers are electrons, 2) the n-type MOS-C curve shape is essentially a mirror image of the p-type MOS-C curve shape, 3) the accumulation region occurs at positive polarities, and 4) the inversion region occurs at negative polarities.

Accumulation region

For a p-type MOS-C, the accumulation region of the C-V curve is observed when negative voltages are applied to the gate. The negative polarity causes majority carriers (holes) to be attracted toward the gate. Because the oxide is a good insulator, these holes accumulate at the substrate-to-oxide/well-to-oxide interface.

A C-V test measures the oxide capacitance in the strong accumulation region, where for a p-type MOS-C, the voltage is negative enough that the capacitance is essentially constant and the C-V curve slope is essentially flat. There, the oxide thickness can be extracted from the oxide capacitance. However, the C-V curve for a very thin oxide often does not saturate to a flat slope. In that case, the measured oxide capacitance differs from the true oxide capacitance.

Depletion region

For a p-type MOS-C, as the gate voltage moves toward positive values, the MOS-C starts to differ from a parallel-plate capacitor. Roughly at the point where the gate voltage becomes positive, the following occurs:

- The positive gate electrostatically repels holes from the substrate-to-oxide/well-to-oxide interface.
- A carrier-depleted area forms beneath the oxide, creating an insulator (recall that the absence of free-moving charges distinguishes an insulator from a conductor).

As a result, the high-frequency Model 4200-CVU measures two capacitances in series: the oxide capacitance and the depletion capacitance. As the gate voltage becomes more positive, the following occurs:

- The depletion zone penetrates more deeply into the semiconductor.
- The depletion capacitance becomes smaller, and consequently, the total measured capacitance becomes smaller.

Therefore, the C-V curve slope is negative in the depletion region.

Inversion region

For a p-type MOS-C, as the gate voltage increases beyond the threshold voltage, dynamic carrier generation and recombination[1] move toward net carrier generation. The positive gate voltage both generates electron-hole pairs and attracts electrons (the minority carriers) toward the gate. Again, because the oxide is a good insulator, these minority carriers accumulate at the substrate-to-oxide / well-to-oxide interface. The accumulated minority-carrier layer is called the inversion layer, because the carrier polarity is inverted. Above a certain positive gate voltage, most available minority carriers are in the inversion layer, and further gate-voltage increases do not further deplete the semiconductor. That is, the depletion region reaches a maximum depth.

However, inversion-charge generation is slower than the 1 MHz or 100 kHz frequency of the HF-CV (High Frequency-CV) measurement. The average time to generate an inversion charge is $\sim 10\tau_g N_a/n_i$, where τ_g is the generation lifetime (seconds), N_a is the doping concentration (cm^{-3}), and n_i is the intrinsic carrier concentration (cm^{-3}). For a $10^{15}\ cm^{-3}$ doping concentration and microsecond generation lifetime, electron-hole-pair (ehp) generation cannot keep up with the high frequency measurement signal. Therefore, once the depletion region reaches a maximum depth, the capacitance that is measured by the HF-CV analyzer is still based on the majority carrier position and distribution. The following applies:

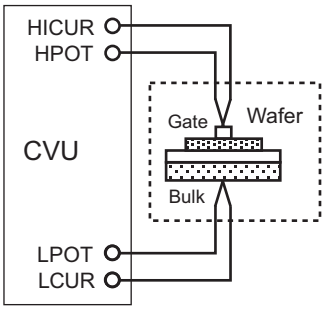
- The capacitance that is measured by the HF-CV analyzer is the oxide capacitance in series with maximum depletion capacitance. This capacitance is often referred to as minimum capacitance.
- The C-V curve slope is almost flat.

NOTE *The measured inversion-region capacitance at the maximum depletion depth depends on the measurement frequency. Therefore, C-V curves measured at different frequencies may have different appearances. Generally, such differences are more significant at lower frequencies and less significant at higher frequencies.*

Project plan tests

The following tests using the Model 4200-CVU for MOS capacitor testing are run from the following project plan: CVU_MOScap. Figure 15-106 shows the basic test configuration for testing a MOS capacitor.

Figure 15-106
Simplified configuration to test MOS capacitor



CVSweep_MOScap ITM

This test performs a capacitance measurement at each step of a user-configured linear voltage sweep. Using the acquired C-V data, the formulator calculates several device parameters

1. Although the average net concentration of carriers in a semiconductor is stable at equilibrium, carrier generation and recombination occur dynamically.

including oxide capacitance (COX), oxide thickness (TOX), flatband capacitance (CFB), flatband voltage (VFB), doping density (NDOPING), depletion depth (DEPTHM), Debye length (DEBYEM), threshold voltage (VTH), and the effective oxide charge (QEFF).

The series resistance (RS) is also calculated from the capacitance (in strong accumulation) and the conductance. The corrected capacitance (CADJ) is calculated by compensating for the series resistance.

C-2vsV_MOScap ITM

This test performs a C-V sweep and displays the capacitance, $1/C^2$, as a function of the gate voltage. A sweep can yield important information about doping profile. In the following equation, *N* (NDOPING) is related to the reciprocal of the slope of the $1/C^2$ versus V_G curve:

$$N = \left| \frac{-2}{q\varepsilon_S A^2 \frac{d(1/C^2)}{dV}} \right| \quad \textit{Eq. 1}$$

Where: N = Doping concentration (cm^{-3})
A = Gate area (cm^2)
C = Measured capacitance (F)
ε_S = Permittivity of the substrate material (F/cm)
q = Electron charge (1.60219×10^{-19} C)
V = Gate voltage

A positive slope indicates acceptors and a negative slope indicates donors.

The substrate doping concentration is extracted from the slope of the $1/C^2$ curve and is displayed on the graph.

DopingProfile_MOScap ITM

This test performs C-V measurements and graphically displays the doping concentration (*N*) as a function of depth (*W*). The doping concentration (*N*) is calculated above. The depletion depth (*W*) - called DEPTHM in the formulator - is computed from the high frequency capacitance and the oxide capacitance at each measured value of the gate voltage (from Nicollian and Brews, p. 186; see References). The program computes each *W* element of the calculated data array as shown below:

$$W = A\varepsilon_S\left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2}) \quad \textit{Eq. 2}$$

Where: W = Depth (m)
A = Gate area (cm^2)
C = Measured capacitance (F)
ε_s = Permittivity of the substrate material (F/cm)
C_{OX} = Oxide capacitance (F)
$1 \times E^{-2}$ = Units conversion from cm to m

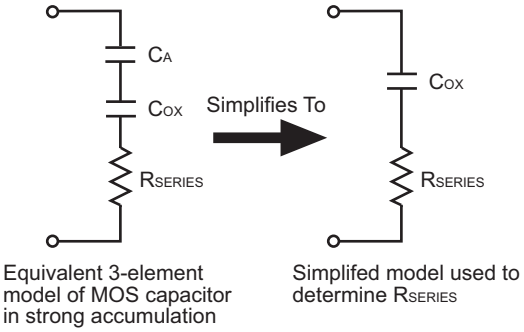
Compensating for series resistance

After generating a C-V curve, measurements may need to be compensated for series resistance. The series resistance (R_{SERIES}) can be attributed to either the substrate (well) or the backside of the wafer. For wafers typically produced in fabs, the substrate bulk resistance is fairly small (<10 Ω) and has negligible impact on C-V measurements. However, if the backside of the wafer is used as an electrical contact, the series resistance due to oxides can significantly distort a measured C-V curve.

Without series compensation, capacitance can be lower than normal, and C-V curves can be distorted. Tests for this project compensate for series resistance using the simplified 3-element model shown in Figure 15-107.

In this model, C_{OX} is the oxide capacitance while C_A is the capacitance of the accumulation layer. The series resistance is represented by R_{SERIES}.

Figure 15-107
Simplified model to determine series resistance



The corrected capacitance (C_{ADJ}) and corrected conductance (G_{ADJ}) are calculated from the formulas shown below (from Nicollian and Brews, p. 224; see References):

$$C_{ADJ} = \frac{(G^2 + (2\pi fC)^2)C}{a_R{}^2 + (2\pi fC)^2} \quad \textit{Eq. 4}$$

$$G_{ADJ} = \frac{(G^2 + (2\pi fC)^2)Q_R}{a_R{}^2 + (2\pi fC)^2} \quad \textit{Eq. 5}$$

where $a_R = G - (G^2 + (2\pi fC)^2)R_S$ *Eq. 6*

Where: C_{ADJ} = Series resistance compensated parallel model capacitance
G_{ADJ} = Series resistance compensated conductance
C = Measured parallel model capacitance
G = Measured conductance
f = test frequency as set in KITE (Definition tab)
R_S = Series resistance

The series resistance, R_S, may be calculated from the capacitance and conductance values that are measured while biasing the DUT (device under test) in the accumulation region. The series resistance is calculated as follows:

$$R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1 + \left(\frac{G}{2\pi fC}\right)^2\right]G} \quad \textit{Eq. 7}$$

Where: R_S = Series resistance
G = Measured conductance
C = Measured parallel model capacitance (in strong accumulation)
f = test frequency as set in KITE (Definition tab)

NOTE *The above equations for compensating for series resistance require that the Model 4200-CVU is using the parallel model (Cp-Gp).*

For this project, these formulas have been added into the KITE formulator so the capacitance and conductance can be automatically compensated for the series resistance.

Extracting MOS device parameters from C-V measurements

Oxide thickness

For a relatively thick oxide (>50Å), extracting the oxide thickness is fairly simple. The oxide capacitance (C_{OX}) is the high frequency capacitance when the device is biased for strong accumulation. In the strong accumulation region, the MOS-C acts like a parallel-plate capacitor, and the oxide thickness (T_{OX}) may be calculated from C_{OX} and the gate area using the following equation:

$$T_{OX} = \frac{1xE^{7}A\varepsilon_{OX}}{C_{OX}} \qquad \textit{Eq. 8}$$

Where: T_{OX} = Oxide thickness (nm)
A = Gate area (cm^2)
ε_{OX} = Permittivity of the oxide material (F/cm)
C_{OX} = Oxide capacitance (F)
$1 \times E^7$ = Units conversion from cm to nm

NOTE *Oxide thickness calculations based on C-V measurements can be very precise, unless improperly extracted. Oxide thickness must be extracted from the strong accumulation region, where the capacitance measured truly reflects the oxide capacitance.*

The oxide capacitance, C_{OX}, is set in the formulator to be the maximum capacitance in accumulation.

Flatband capacitance and flatband voltage

Application of a certain gate voltage, the flatband voltage (V_{FB}) results in the disappearance of band bending. At this point, known as the flatband condition, the semiconductor band is said to become flat. Because the band is flat, the surface potential is zero (with the reference potential being taken as the bulk potential deep in the semiconductor). Flatband voltage and its shift are widely used to extract other device parameters, such as oxide charges.

V_{FB} can be identified from the C-V curve. One way is to use the flatband capacitance method. For this method, the ideal value of the flatband capacitance (C_{FB}) is calculated using equations (9) and (10). Once the value of C_{FB} is known, the value of V_{FB} can be obtained from the C-V curve data, by interpolating between the closest gate-to-substrate (V_{GS}) values (from Nicollian and Brews pp 487-488; see References).

Equation (10) calculates the Debye length parameter (λ) that is used in equation (9). Based on the doping profile, the λ calculation requires one of the following doping concentrations: N at 90% of W_{MAX} (refer to Nicollian and Brews), a user-supplied N_A (bulk doping concentration for a p-type, acceptor, material), or a user-supplied N_D (bulk doping concentration for an n-type, donor, material).

NOTE *The flatband capacitance method is invalid when the interface trap density (D_{IT}) becomes very large (10^{12} - 10^{13} or greater). However, the method should give satisfactory results for most users. Those dealing with high D_{IT} values should consult the appropriate literature for a more suitable method.*

Use equations (9) and (10) to calculate the flatband capacitance:

$$C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)}{C_{OX} + \left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)} \quad \textit{Eq. 9}$$

Where: C_{FB} = Flatband capacitance (F)
C_{OX} = Oxide capacitance (F)
ε_S = Permittivity of the substrate material (F/cm)
A = Gate area (cm^2)
$1 \times E^2$ = Units conversion from m to cm
λ = Extrinsic Debye length, which is calculated as follows:

$$\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2}(1xE^{-2}) \quad \textit{Eq. 10}$$

Where: λ = Extrinsic Debye length
ε_S = Permittivity of the substrate material (F/cm)
kT = Thermal energy at room temperature (293K) (4.046×10^{-21}J)
q = Electron charge (1.60219×10^{-19} C)
N_X = N at 90% W_{MAX} or N90W (refer to Nicollian and Brews; see References) or, when input by the user, $N_X = N_A$ or $N_X = N_D$
$1 \times E^{-2}$ = Units conversion from cm to m

The extrinsic Debye length is an idea borrowed from plasma physics. In semiconductors, majority carriers can move freely. The motion is similar to a plasma. Any electrical interaction has a limited range. The Debye length is used to represent this interaction range. Essentially, the Debye length indicates how far an electrical event can be sensed within a semiconductor.

Threshold voltage

The turn-on region for a MOSFET corresponds to the inversion region on its C-V plot. When a MOSFET is turned on, the channel formed corresponds to strong generation of inversion charges. It is these inversion charges that conduct current. When a source and drain are added to a MOS-C to form a MOSFET, a p-type MOS-C becomes an n-type MOSFET, also called an n-channel MOSFET. Conversely, an n-type MOS-C becomes a p-channel MOSFET.

The threshold voltage (V_{TH}) is the point on the C-V curve where the surface potential (ϕ_S) equals twice the bulk potential (ϕ_B). This curve point corresponds to the onset of strong inversion. For an enhancement-mode MOSFET, V_{TH} corresponds to the point where the device begins to conduct. The physical meaning of the threshold voltage is the same for both a MOS-C C-V curve and a MOSFET I-V curve. However, in practice, the numeric V_{TH} value for a MOSFET may be slightly different, due to the particular method used to extract the threshold voltage.

The threshold voltage of a MOS capacitor can be calculated as follows:

$$V_{TH} = V_{FB} \pm \left[\frac{A}{C_{OX}}\sqrt{4\varepsilon_S q |N_{BULK}\phi_B|} + 2|\phi_B|\right] \quad \textit{Eq. 11}$$

Where: V_{TH} = Threshold voltage (V)
V_{FB} = Flatband potential (V)
A = Gate area (cm^2)
C_{OX} = Oxide capacitance (F)
ε_S = Permittivity of the substrate material (F/cm)
q = Electron charge (1.60219×10^{-19} C)

N_{BULK} = Bulk doping (cm^{-3}) (Note: The formulator name for N_{BULK} is N90W)
ϕ_B = Bulk potential (V) (Note: The formulator name for ϕ_B is PHIB)

The bulk potential is calculated as follows:

$$\phi_B = -\frac{kT}{q}\ln\left(\frac{N_{BULK}}{N_i}\right)(DopeType) \quad \textit{Eq. 12}$$

Where: ϕ_B = Bulk potential (V) (Note: The formulator name for ϕ_B is PHIB)
k = Boltzmann's constant (1.3807 x 10^{-23} J/K)
T = Test temperature (K)
q = Electron charge (1.60219 x 10^{-19} C)
N_{BULK} = Bulk doping (cm^{-3}) (Note: The formulator name for N_{BULK} is called N90W)
N_i = Intrinsic carrier concentration (1.45 x 10^{10} cm^{-3})
DopeType = +1 for p-type materials and -1 for n-type materials (Note: The value for *DopeType* is changed in the Constants area of the formulator.

Metal-semiconductor work function difference

The metal-semiconductor work function difference (W_{MS}) is commonly referred to as the work function. It contributes to the shift in V_{FB} from the ideal zero value, along with the effective oxide charge (Nicollian and Brews, pp. 462-477, Sze, pp. 395-402; see References). The work function represents the difference in work necessary to remove an electron from the gate and from the substrate. The work function is derived as follows:

$$W_{MS} = W_M - \left[W_S + \frac{E_{BG}}{2} - \phi_B\right] \quad \textit{Eq. 13}$$

Where: W_{MS} = Work function
W_M = Metal work function (V) *
W_S = Substrate material work function, electron affinity (V) *
E_{BG} = Substrate material bandgap (V) *
ϕ_B = Bulk potential (V) (Note: The formulator name for ϕ_B is PHIB)

* The values for W_M, W_S, and E_{BG} are listed in the formulator as constants. You can change the values depending on the type of materials.

The following example calculates the work function for silicon, silicon dioxide, and aluminum:

$$W_{MS} = 4.1 - \left[4.15 + \frac{1.12}{2} - \phi_B\right] \quad \textit{Eq. 14}$$

Therefore,

$$W_{MS} = -0.61 + \phi_B \quad \textit{Eq. 15}$$

and

$$W_{MS} = -0.61 - \left(\frac{kT}{q}\right)\ln\left(\frac{N_{BULK}}{n_1}\right)(DopeType) \quad \textit{Eq. 16}$$

Where: W_{MS} = Work function
k = Boltzmann's constant (1.3807 x 10^{-23} J/K)
T = Test temperature (K)
q = Electron charge (1.60219 x 10^{-19} C)
N_{BULK} = Bulk doping (cm^{-3})
DopeType = +1 for p-type materials and -1 for n-type materials (Note: The value for *DopeType* is changed in the Constants area of the formulator).

For example, for an MOS capacitor with an aluminum gate and p-type silicon (N_{BULK} = 10^{16} cm^{-3}), W_{MS} = -0.95 V. Also, for the same gate and n-type silicon (N_{BULK} = 10^{16} cm^{-3}), W_{MS} = -0.27V.

Because the supply voltage of modern CMOS devices is decreasing and since aluminum reacts with silicon dioxide, heavily doped polysilicon is often used as the gate material. The goal is to achieve a minimal work-function difference between the gate and the semiconductor, while maintaining the conductive properties of the gate.

Effective and total bulk oxide charge

The effective oxide charge, Q_{EFF}, represents the sum of oxide fixed charge (Q_F), mobile ionic charge (Q_M), and oxide trapped charge (Q_{OT}):

$$Q_{EFF} = Q_F + Q_M + Q_{OT} \quad \textit{Eq. 17}$$

Q_{EFF} is distinguished from interface trapped charge (Q_{IT}), in that Q_{IT} varies with gate bias and Q_{EFF} does not (Nicollian and Brews pp. 424-429, Sze pp. 390-395; See References). Simple measurements of oxide charge using C-V measurements do not distinguish the three components of Q_{EFF}. These three components can be distinguished from one another by temperature cycling, as discussed in Nicollian and Brews, p. 429, Fig. 10.2. Also, since the charge profile in the oxide is not known, the quantity, Q_{EFF}, should be used as a relative, not absolute measure of charge. It assumes that the charge is located in a sheet at the silicon-silicon dioxide interface.

From Nicollian and Brews, Eq. 10.10, we have:

$$V_{FB} - W_{MS} = -\frac{Q_{EFF}}{C_{OX}} \quad \textit{Eq. 18}$$

Where: V_{FB} = Flatband potential (V)
W_{MS} = Metal-semiconductor work function (V)
Q_{EFF} = Effective oxide charge (C)
C_{OX} = Oxide capacitance (F)

Note that C_{OX} here is per unit of area. So that:

$$Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A} \quad \textit{Eq. 19}$$

Where: Q_{EFF} = Effective oxide charge (C)
C_{OX} = Oxide capacitance (F)
W_{MS} = Metal-semiconductor work function (V)
V_{FB} = Flatband potential (V)
A = Gate area (cm^{-2})

For example, assume a 0.01cm^2, 50 pF, p-type MOS-C with a flatband voltage of –5.95 V; its N_{BULK} of 10^{16} cm^{-3} corresponds to a W_{MS} of –0.95 V. For this example, Q_{EFF} calculates to be 2.5 x 10^{-8} C/cm^2, which then causes the threshold voltage to shift ~ 5 V in the negative direction. Note that in most cases where the bulk charges are positive, there is a shift toward negative gate voltages. The effective oxide charge concentration (N_{EFF}) is computed from effective oxide charge (Q_{EFF}) and the electron charge as follows:

$$N_{EFF} = \frac{Q_{EFF}}{q} \quad \textit{Eq. 20}$$

Where: N_{EFF} = Effective oxide charge density (cm^2)
Q_{EFF} = Effective oxide charge (C)
q = Electron charge (1.60219 x 10^{-19} C)

References

Nicollian, E.H. and Brews, J.R., MOS Physics and Technology, Wiley, New York (1982).
Sze, S.M., Physics of Semiconductor Devices, 2nd edition. Wiley, New York (1985).

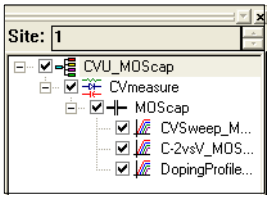
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_MOScap** folder.
5. Open the **CVU_MOScap.kpr** project.

The CVU_MOScap project plan is shown in Figure 15-108.

Figure 15-108
CVU_MOScap project plan

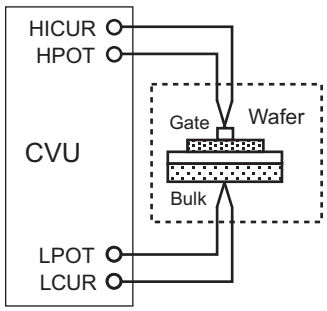


Connections

Figure 15-109 shows the basic test configuration. Connections provides details on connections to a semiconductor wafer. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-109
Basic configuration to test a MOS capacitor



Formulas and constants

Formulas and user-defined constants that are used for the application tests are summarized (in alphabetical order) in Table 15-23 (formulas) and Table 15-24 (constants).

Table 15-23
Formulas for the CVU_MOSCap project

Formula Name	Formula Details	
AR (a_R)	Units: none	Description: Intermediate parameter for calculation of corrected capacitance
	AR = GP_GB-(GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*RS	
	Simplified Equation: $a_R = G - (G^2 + (2\pi fC)^2)R_S$ See Eq. 6 for details*	
BEST_HI	Units: none	Description: Index from DEPTHM array that is 95% of maximum depletion length, or twice the screening length in the semiconductor, whichever is larger
	BEST_HI = FINDD(DEPTHM,COND(2*DEBYEM*SQRT(LN(ABS(N90W/NI))), MAX(DEPTHM), 2*DEBYEM*SQRT(LN(ABS(N90W/NI)))), 0.95*MAX(DEPTHM), 2)	
BEST_LO	Units: none	Description: Index from DEPTHM array that is three Debye lengths from the surface
	BEST_LO = FINDD(DEPTHM, 3*DEBYEM, 2)	
CADJ (C_{ADJ})	Units: F	Description: Corrected capacitance by compensating series resistance
	CADJ = ((GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*(CP_GB)/(AR^2 + (2*PI*F_GB*CP_GB)^2)	
	Simplified Equation: $C_{ADJ} = \frac{(G^2 + (2\pi fC)^2)C}{a^2 + (2\pi fC)^2}$ See Eq. 4 for details*	
CFB (C_{FB})	Units: F	Description: Flatband capacitance
	CFB = (COX*ES*AREA/(DEBYEM*1E2))/(COX+(ES*AREA/(DEBYEM*1E2)))	
	Simplified Equation: $C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)}{C_{OX} + \left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)}$ See Eq. 9 for details*	
COX (C_{OX})	Units: F	Description: Oxide capacitance (usually set to maximum capacitance in accumulation)
	COX = MAX(MAVG(CADJ, 10))+1E-15	
	C_{OX} = Set to the maximum capacitance in accumulation	
CMIN	Units: F	Description: Minimum capacitance
	CMIN = MIN(MAVG(CADJ, 10))+1E-15	

Table 15-23 (continued)
Formulas for the CVU_MOSCap project

DEPYEM (λ)	Units: m	Description: Debye length (in meters)
	DEBYEM = SQRT(ES*K*TEMP/(ABS(N90W)*Q^2))*1E-2	
	Simplified Equation: $\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2} (1xE^{-2})$ See Eq. 10 for details	
DEPTHM (*W*)	Units: m	Description: Depletion depth (in meters)
	DEPTHM = 1E-2*AREA*ES*(1/CADJ-1/COX)	
	Simplified Equation: $W = A\varepsilon_S\left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2})$ See Eq. 2 for details*	
INVCSQR	Units: $1/F^2$	Description: Inverse square of capacitance
	INVCSQR = 1/(MAVG(CADJ, 5))^2	
	Simplified Equation: $INVCSQR = \frac{1}{C^2}$	
MAXINVSQR	Units: $1/F^2$	Description: Finds row position of maximum point on $1/C^2$ curve
	MAXINVSQR = MAXPOS(INVCSQR)	
N90W	Units: none	Description: Doping density at 90% of maximum depletion depth
	N90W = AT(NDOPING, FINDD(DEPTHM, 0.9*MAX(DEPTHM), 2))	
NAVG (N_{AVG})	Units: none	Description: Average doping calculated between index BEST_HI and BEST_LO
	NAVG = AVG(SUBARRAY(NDOPING, COND(BEST_HI, BEST_LO, BEST_HI, BEST_LO), COND(BEST_HI, BEST_LO, BEST_LO, BEST_HI))))	
NDOPING (*N*)	Units: $1/cm^3$	Description: Doping density
	NDOPING = ABS((-2)/(AREA^2*Q*ES)/(DELTA(INVCSQR)/DELTA(DCV_AB)))	
	Simplified Equation: $N = \left\lvert \frac{-2}{q\varepsilon_S A^2 \frac{d(1/C^2)}{dV}} \right\rvert$ See Eq. 1 for details*	
NSLOPE	Units: none	Description: Finds slope of $1/C^2$ curve
	NSLOPE = LINFITSLP(DCV_GB, INVCSQR, VFBPOS, MAXINVSQR)	
NSUB	Units:$1/cm^3$	Description: Calculated substrate doping concentration
	NSUB = 2/(NSLOPE*Q*ES*AREA^2)	
PHIB (φ_B)	Units: V	Description: Bulk potential
	PHIB = (-1)*K*TEMP/Q*LN(ABS(N90W)/NI)*DOPETYPE	
	Simplified Equation: $\phi_B = \pm\frac{kT}{q}\ln\left(\frac{N_{BULK}}{N_i}\right)(DopeType)$ See Eq. 12 for details*	

Table 15-23 (continued)
Formulas for the CVU_MOSCap project

QEFF (Q_{EFF})	Units: C/cm^2	Description: Effective oxide charge
	QEFF = COX*(WMS-VFB)/AREA	
	Simplified Equation: $Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A}$ See Eq. 19 and Eq. 17 for details*	
RS (R_S)	Units: Ω	Description: Series resistance calculated from capacitance
	RS = (AT(MAVG(GP_GB, 5)/((2*PI*F_GB)*MAVG(CP_GB, 5)), MAXPOS(MAVG(CP_GB, 5))))^2 ((1+(AT (MAVG(GP_GB, 5)/((2*PI*F_GB)*MAVG(CP_GB, 5)), MAXPOS(MAVG(CP_GB, 5))))^2)*(AT(MAVG (GP_GB, 5),MAXPOS(MAVG(CP_GB, 5)))))	
	Simplified Equation: $R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1 + \left(\frac{G}{2\pi fC}\right)^2\right]G}$ See Eq. 7 for details*	
TOXNM (T_{OX})	Units: nm	Description: Calculated thickness of oxide (in nanometers)
	TOXNM = (1E7*AREA*EOX)/COX	
	Simplified Equation: $T_{OX} = \frac{A\varepsilon_{OX}(1xE^7)}{C_{OX}}$ See Eq. 8 for details*	
VFB (V_{FB})	Units: V	Description: Flatband voltage
	VFB = AT(DCV_AB,FINDD(CADJ, CFB, 2))	
	Once CFB (C_{FB}) is derived, V_{FB} is interpolated from the closest V_{GS} values	
VFBPOS	Units: none	Description: Finds row position of flatband voltage
	VFBPOS = FINDD(DCV_GB, VFB, 2)	
VTH (V_{TH})	Units: V	Description: Threshold voltage
	VTH = VFB+DOPETYPE*(AREA/COX*SQRT(4*ES*Q*ABS(N90W*PHIB)) + 2*ABS(PHIB))	
	Simplified Equation: $V_{TH} = V_{FB} \pm \left[\frac{A}{C_{OX}}\sqrt{4\varepsilon_S q\lvert N_{BULK}\phi_B\rvert} + 2\lvert\phi_B\rvert\right]$ See Eq. 11 for details*	
WMS (W_{MS})	Units: V	Description: Work function difference between metal and semiconductor (WM and WS are defined in the constants area of the formulator)
	WMS = WM-(WS+(EBG/2)-PHIB)	
	Simplified Equation: $W_{MS} = W_M - \left[W_S + \frac{E_{BG}}{2} - \phi_B\right]$ See Eq. 13 for details*	

* Details for referenced equations are found in the documentation for Key concepts for MOScap testing.

Table 15-24
Constants for the CVU_MOSCap project

Constant	Default Value	Units	Description
AREA	10.404 E-03	cm^2	Gate area of device
DOPETYPE	1 E+00	none	1 = P-type, -1 = N-type
EBG	1.12 E+00	eV	E_{BG} - Semiconductor energy gap
EOX	340.0 E-15	F / cm	ε_{OX} - Permittivity of oxide
ES	1.03 E-12	F / cm	ε_S - Semiconductor permittivity
NI	14.5 E+09	cm^{-3}	N_I - Intrinsic carrier concentration
TEMP	293 E+00	K	Test temperature
WM	4.15 E+00	V	W_M - Metal work function
WS	4.05 E+00	V	W_S - Silicon electron affinity

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

CVSweep_MOScap ITM

Test summary

This ITM performs a capacitance measurement at each step of a user-configured linear voltage sweep. From the acquired data, a Capacitance versus Voltage graph is generated and several device parameters are calculated using the formulator. For more information, see Key concepts for MOScap testing.

Parameter settings

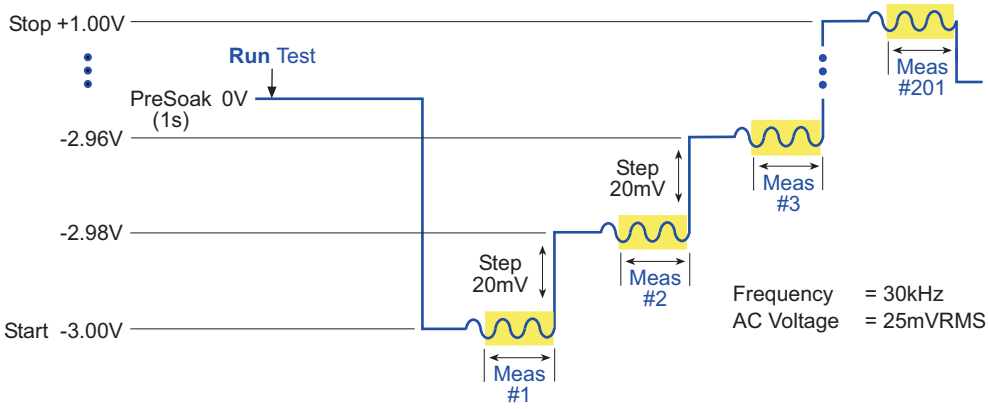
The default parameter settings for this test are listed in Table 15-25. The voltage sweep used for this test is shown in Figure 15-110.

Table 15-25
Parameter settings (CVSweep_MOScap ITM)

Force (Figure 15-112)	**Measure** (Figure 15-112)	**Timing** (Figure 15-111)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -3 V Stop: 1 V Step: 0.02 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 25 mV Data Points: 201	Cp-Gp DCV Frequency I Range: Auto [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 1 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-112)	**CVU Compensation** [3] (Figure 15-112)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-110
Linear voltage sweep (CVSweep_MOScap ITM)



In the Project Navigator, double-click the **CVSweep_MOScap** ITM to display the Definition tab, which is shown in Figure 15-111.

Figure 15-111
Definition tab (CVSweep_MOScap)

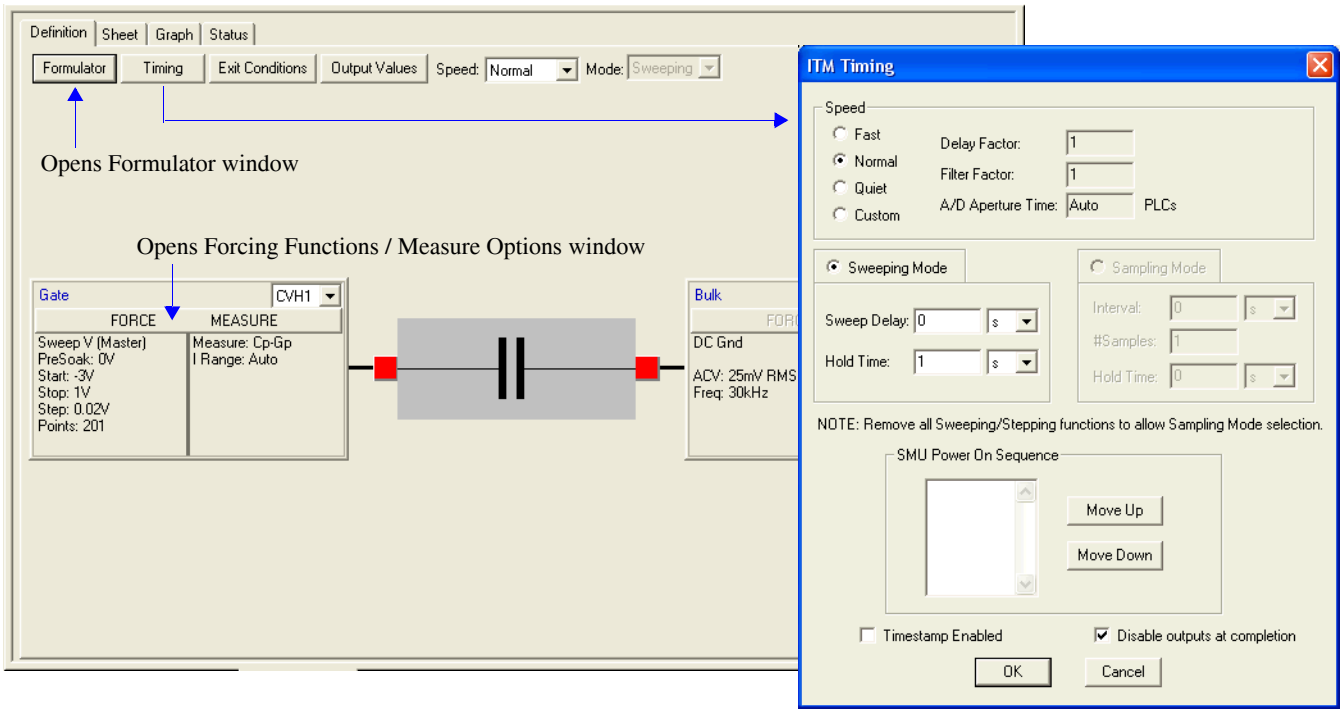
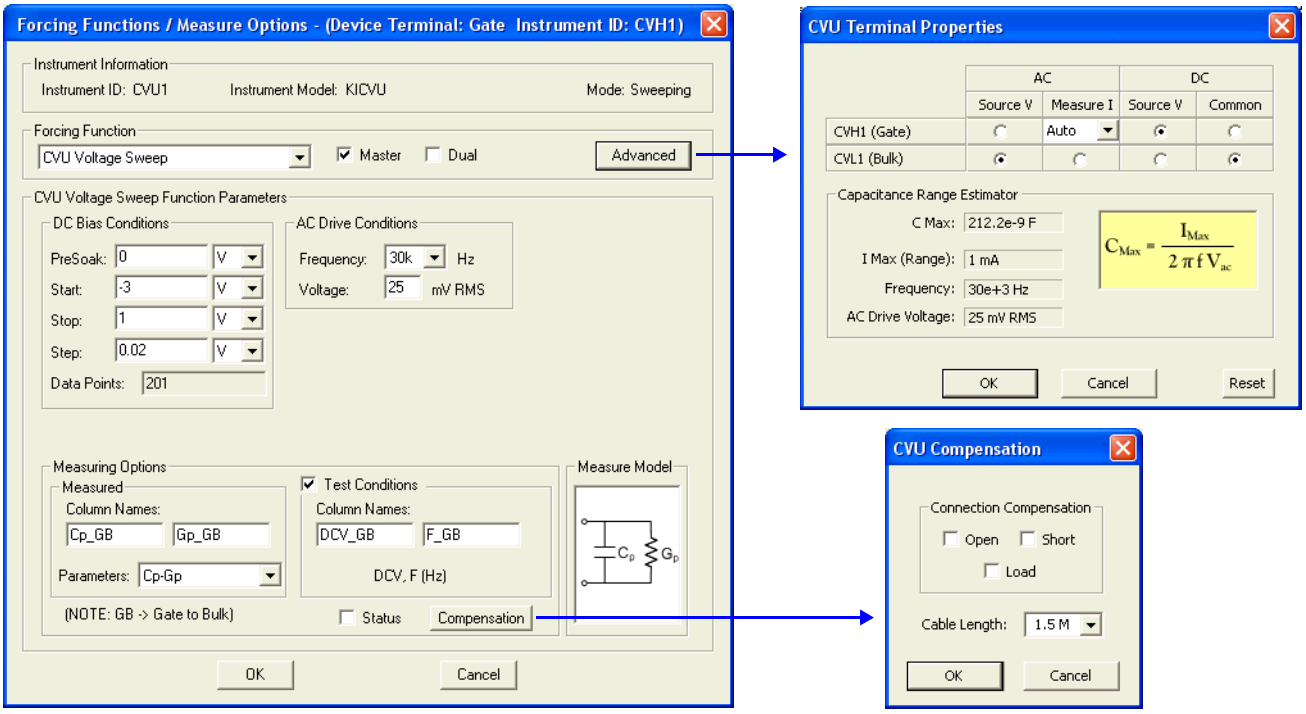


Figure 15-112
Forcing Functions / Measure Options window (CVSweep_MOScap ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

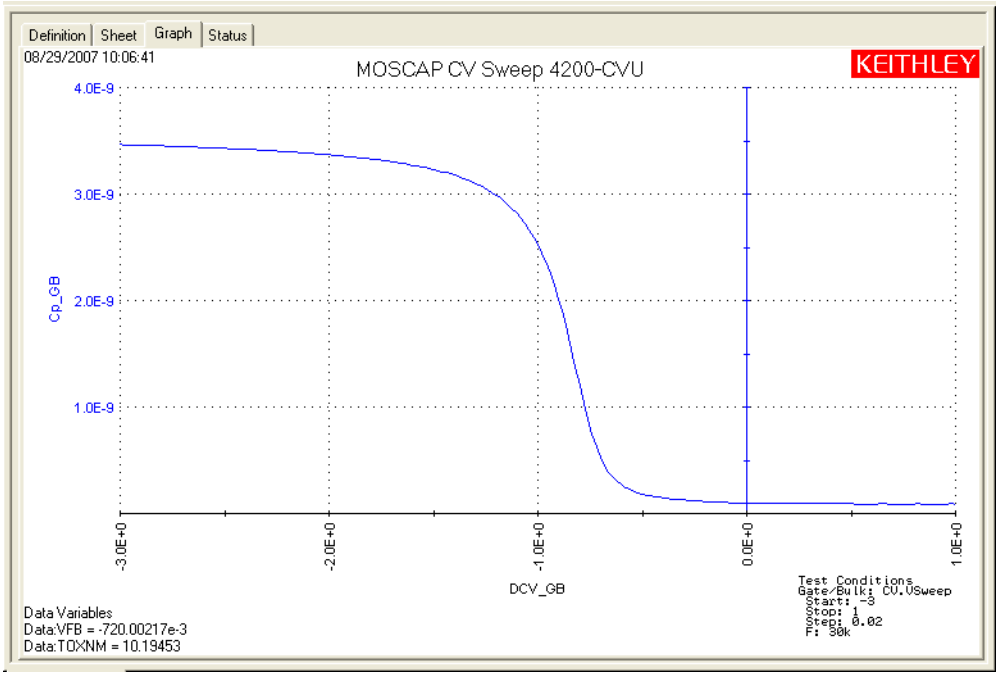
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-113). It includes the results for the VFD and TOXNM formula calculations.

Figure 15-113
Graph tab (CVSweep_MOScap ITM)



C-2vsV_MOScap ITM

Test summary

This ITM performs a C-V sweep and generates a $1/C^2$ versus voltage graph. The graph also shows the doping substrate concentration, which is the result of the NSUB formulator calculation. For more information, see C-2vsV_MOScap ITM.

Parameter settings

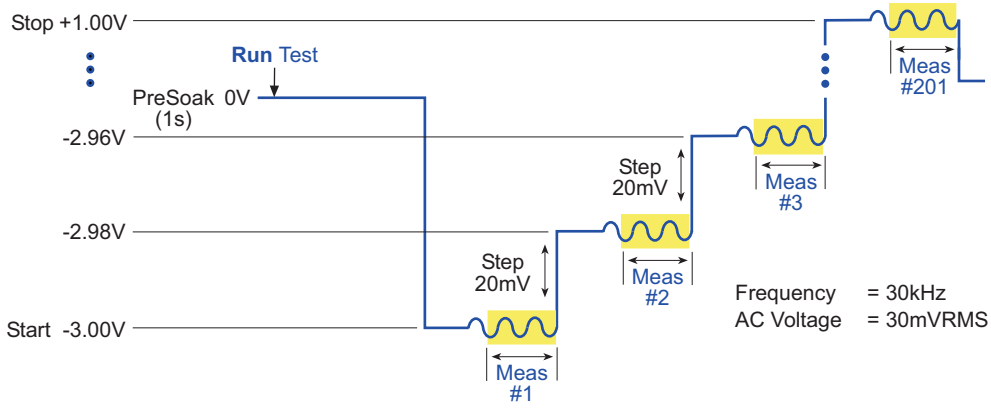
The default parameter settings for this test are listed in Table 15-26. The voltage sweep used for this test is shown in Figure 15-114.

Table 15-26
Parameter settings (C-2vsV_MOScap ITM)

Force (Figure 15-116)	**Measure** (Figure 15-116)	**Timing** (Figure 15-115)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -3 V Stop: 1 V Step: 0.02 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 30 mV Data Points: 201	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 1 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-116)	**CVU Compensation** [3] (Figure 15-116)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-114
Linear voltage sweep (C-2vsV_MOScap ITM)



In the Project Navigator, double-click the **C-2vsV_MOScap** ITM to display the Definition tab, which is shown in Figure 15-115.

Figure 15-115
Definition tab (C-2vsV_MOScap)

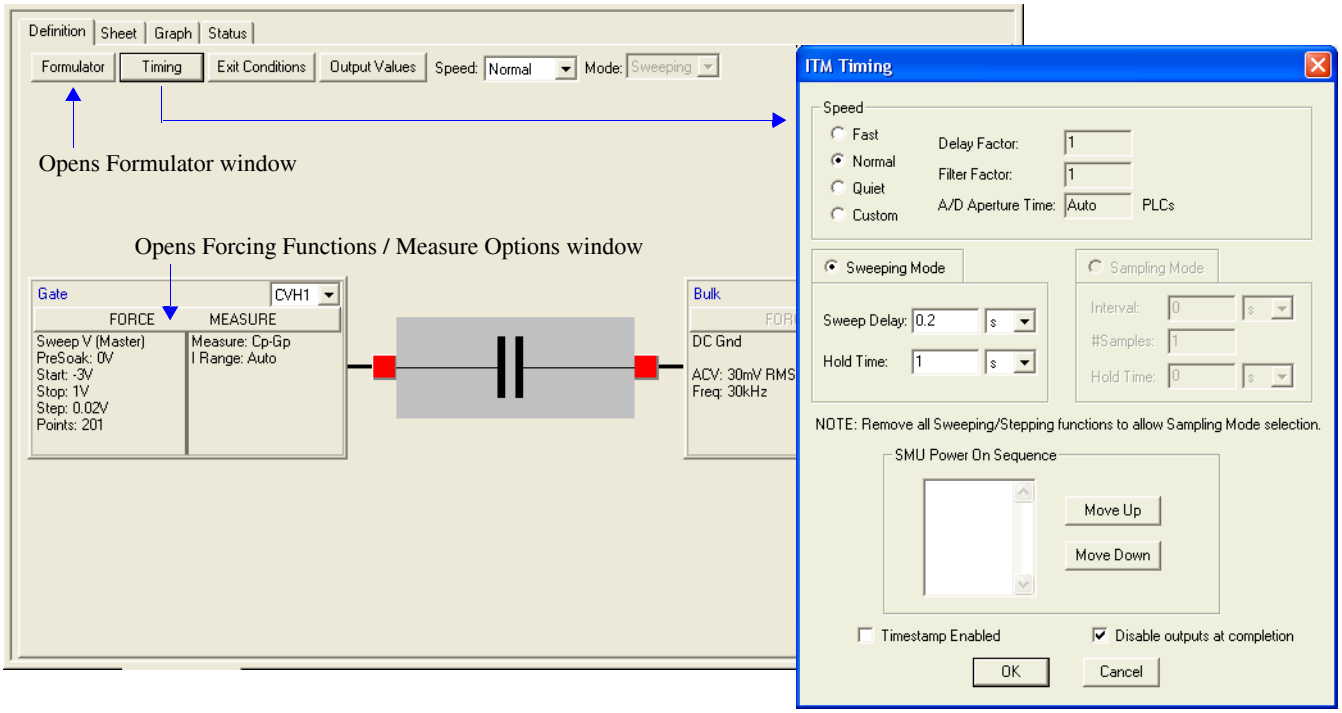
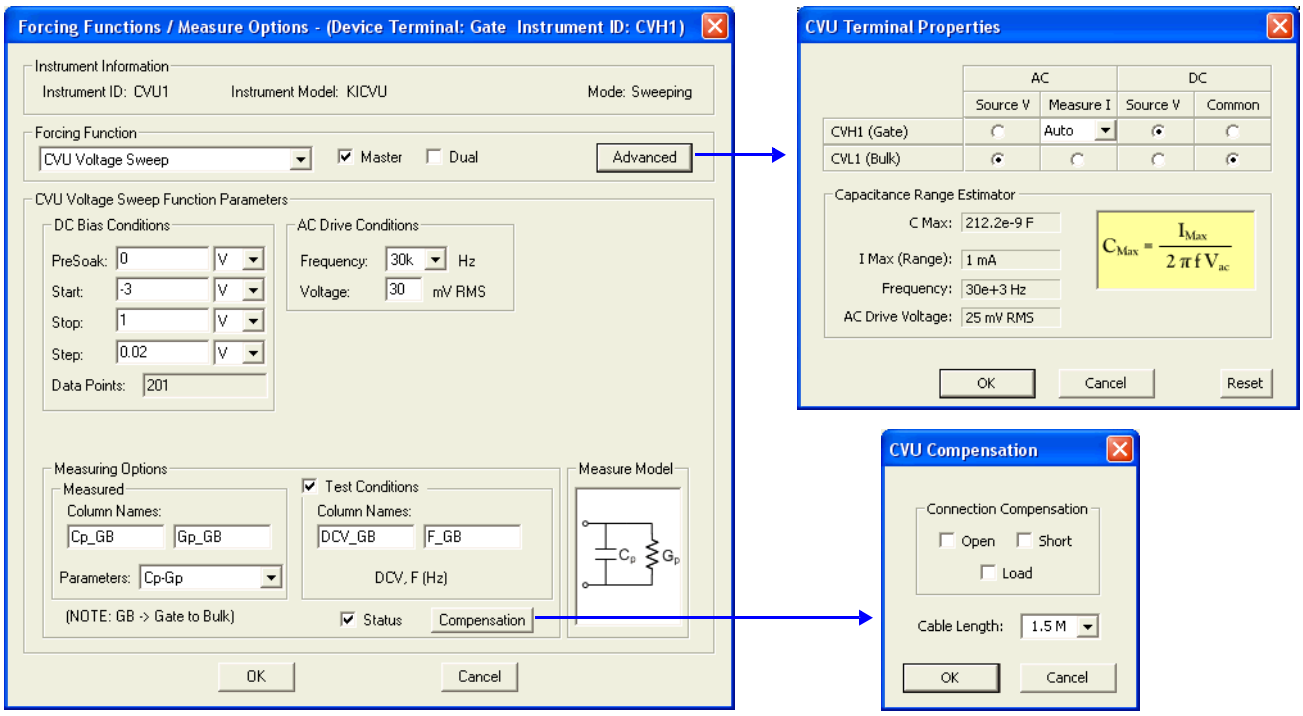


Figure 15-116
Forcing Functions / Measure Options window (C-2vsV_MOScap ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

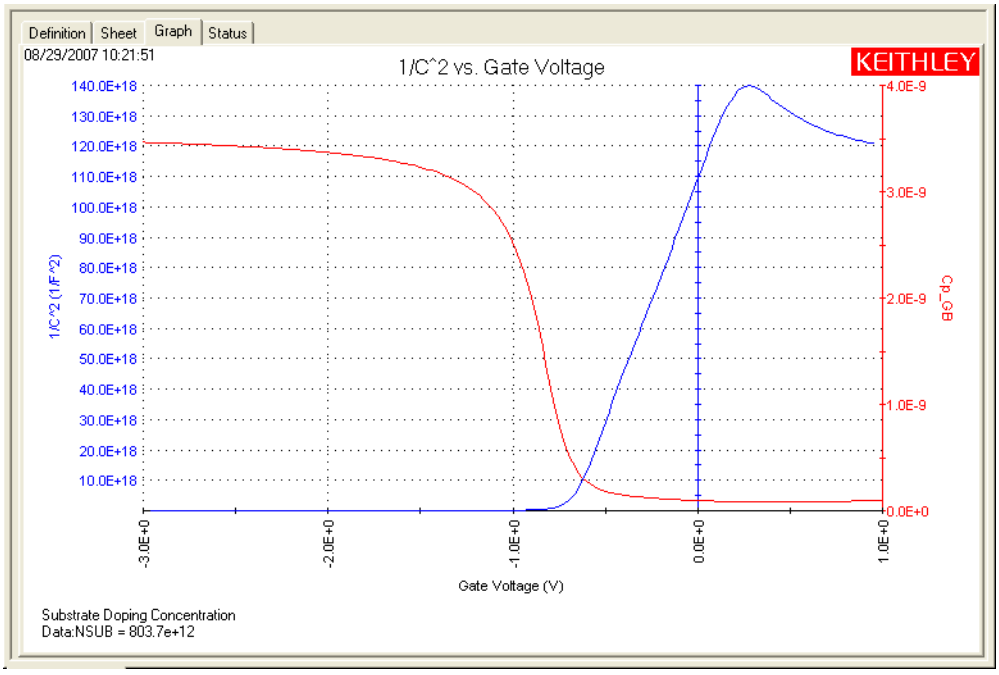
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-117). It includes the result for the NSUB calculation.

Figure 15-117
Graph tab (C-2vsV_MOScap ITM)



DopingProfile_MOSC ITM

Test summary

This ITM performs C-V measurements and graphically displays the doping concentration (*N*) as a function of depth (*W*). For more information, see DopingProfile_MOScap ITM.

Parameter settings

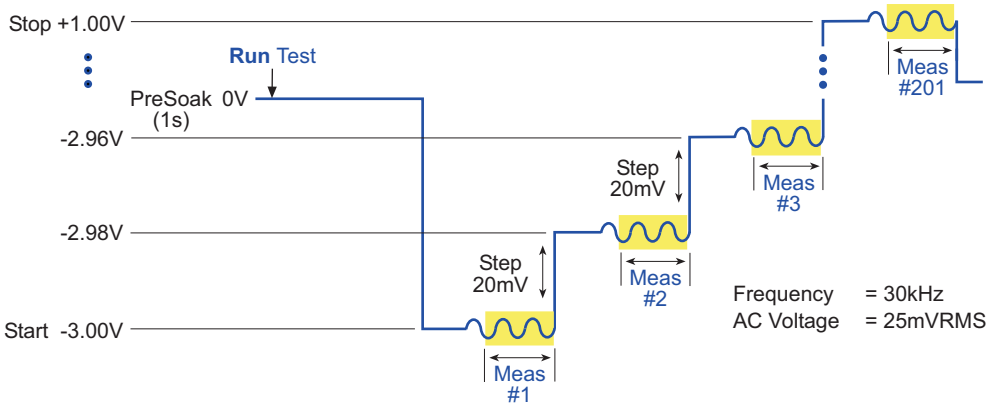
The default parameter settings for this test are listed in Table 15-27. The voltage sweep used for this test is shown in Figure 15-118.

Table 15-27
Parameter settings (DopingProfile_MOSC ITM)

Force (Figure 15-120)	**Measure** (Figure 15-120)	**Timing** (Figure 15-119)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -3 V Stop: 1 V Step: 0.02 V AC Drive Conditions Frequency: 30 kHz AC Voltage: 25 mV Data Points: 201	Cp-Gp DCV Frequency I Range: 30µA [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 1 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-120)	**CVU Compensation** [3] (Figure 15-120)	
Gate: DC Source V Gate: AC Measure I (30µA)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-118
Linear voltage sweep (DopingProfile_MOSC ITM)



In the Project Navigator, double-click the **DopingProfile_MOScap** ITM to display the Definition tab, which is shown in Figure 15-119.

Figure 15-119
Definition tab (DopingProfile_MOSC)

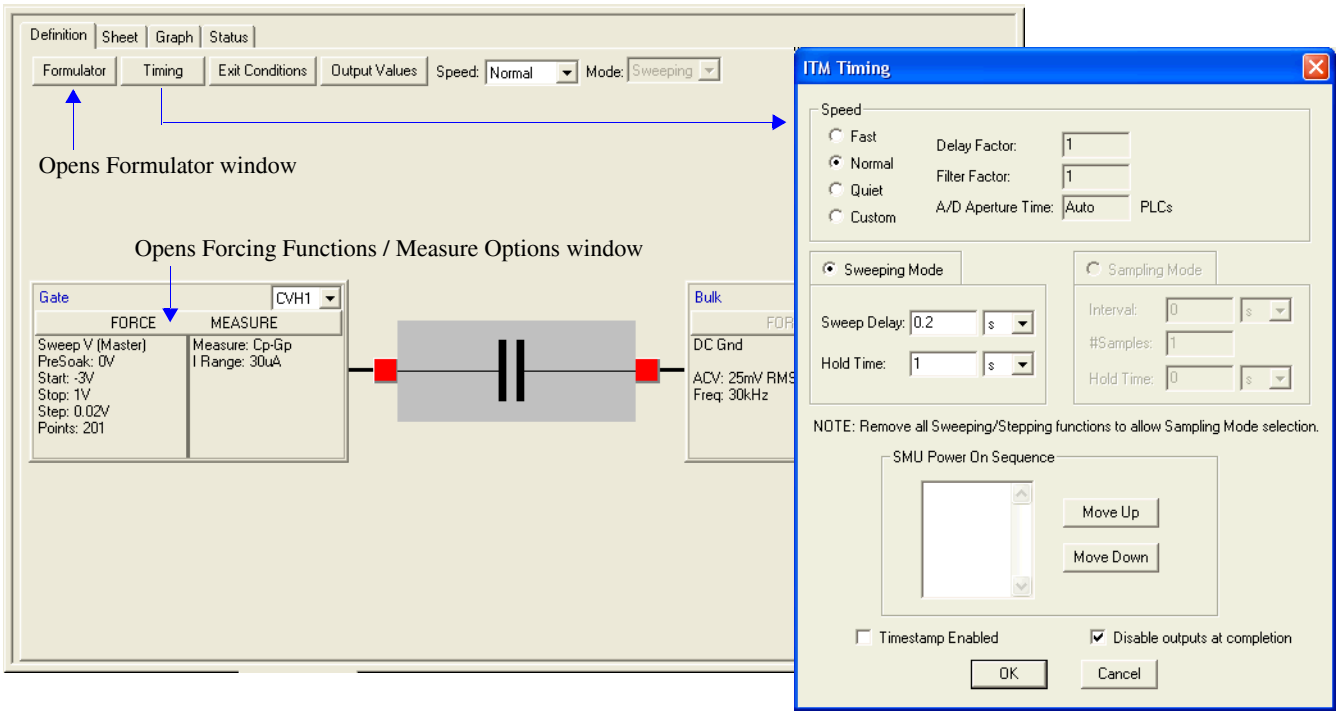
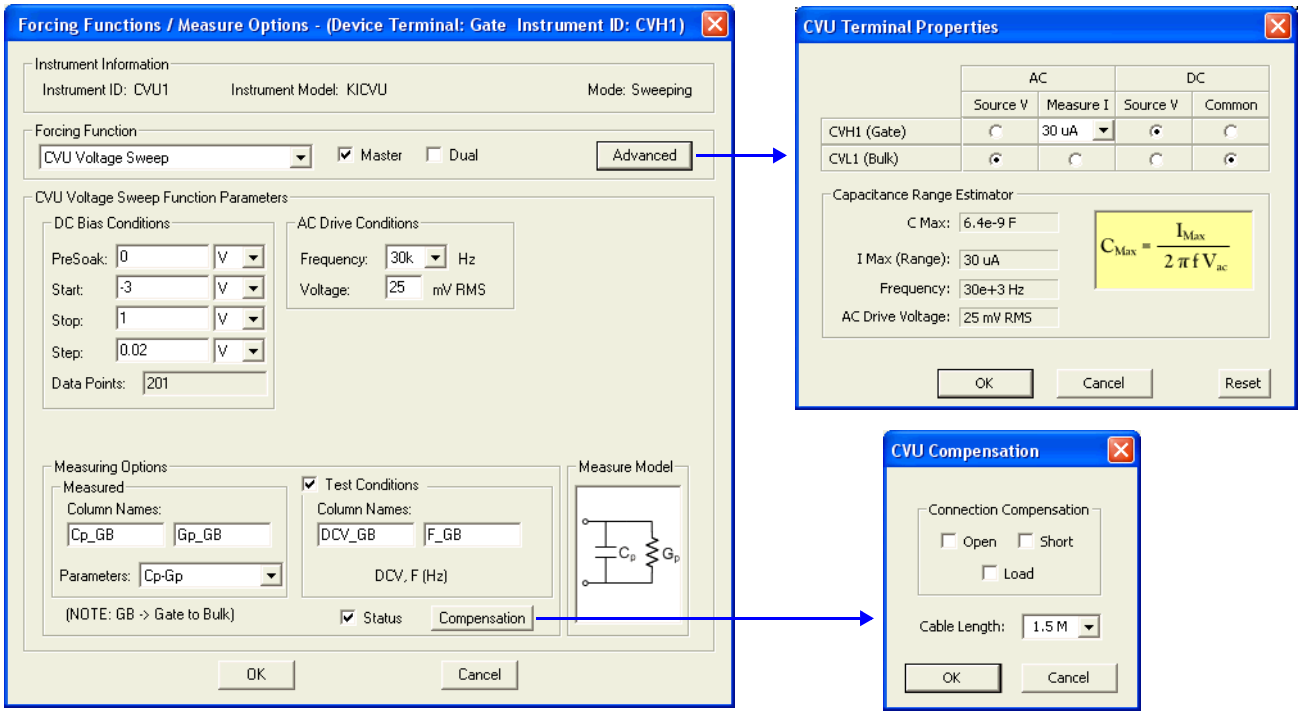


Figure 15-120
Forcing Functions / Measure Options window (DopingProfile_MOSC ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

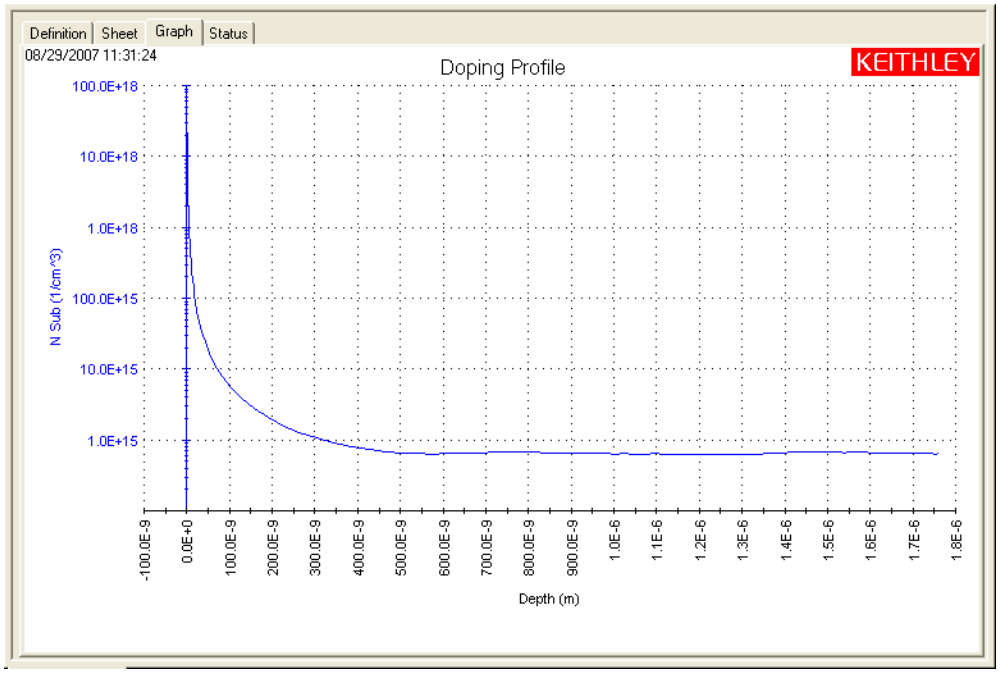
- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graph

The graph is displayed in the Graph tab (see Figure 15-121).

Figure 15-121
Graph tab (DopingProfile_MOSC ITM)



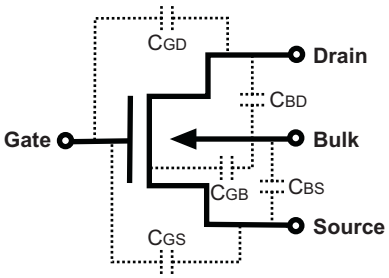
CVU_MOSFET

Project Plan

Key concepts for MOSFET testing

Capacitance measurements are often made on MOSFETs to explore the basic operation and various parameters of the device. Since the high-frequency operation and switching speed of a MOSFET is dependent on the capacitance of the device, capacitance measurements are often made to various parts of the device as shown in Figure 15-122. For example, the capacitance between the gate and channel (C_{GD} and C_{GS}) is important because it creates the charges necessary for operating the devices. This gate-channel capacitance depends on the applied voltage and the operating region.

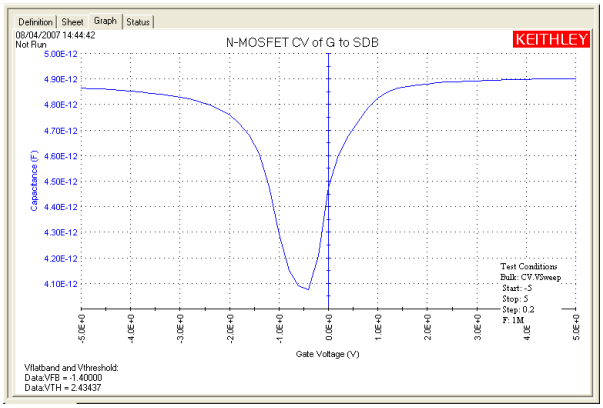
Figure 15-122
N-MOSFET with terminal capacitances



C-V measurements are often made on MOSFETs to extract particular device parameters, such as oxide capacitance (C_{OX}), flatband capacitance (C_{FB}), and oxide thickness (T_{OX}).

The G_to_SDB ITM plots the capacitance as a function of the gate voltage from the two terminal measurement. A typical two-terminal C-V curve of an N-channel MOSFET is shown in Figure 15-123.

Figure 15-123
CVU_MOScap project plan



Notice from the high frequency curve that when the device is in the inversion region, the capacitance is high, unlike the MOS capacitor, which has low capacitance in inversion. This is because the MOSFET has a source and drain, which enables inversion charge to flow, unlike the MOS capacitor, which relies on generation and recombination in the bulk region.

Oxide capacitance

The oxide capacitance (C_{OX}) is usually set to the maximum capacitance in accumulation, and is calculated by the COX formula in the formulator.

Oxide thickness

$$T_{OX} = \frac{1xE^{7} A \varepsilon_{OX}}{C_{OX}}$$

Where: T_{OX} = Oxide thickness (nm)
A = Gate area (cm^2)
ε_{OX} = Permittivity of the oxide material (F/cm)
C_{OX} = Oxide capacitance (pF)
$1 \times E^{7}$ = Units conversion from cm to nm
λ = Extrinsic Debye length

Oxide thickness is calculated by the TOX formula in the formulator.

Depth

The DopingProfile ITM performs a C-V sweep on the two-terminal MOSFET. The doping concentration (N) is calculated and plotted as a function of depletion depth.

$$W = A\varepsilon_{S}\left(\frac{1}{C} - \frac{1}{C_{OX}}\right)(1xE^{-2})$$

Where: W = Depletion depth (m)
A = Gate area (cm^2)
C = Measured capacitance (F)
ε_s = Permittivity of the substrate material (F/cm)
C_{OX} = Oxide capacitance (F)
$1 \times E^{-2}$ = Units conversion from cm to m

Depletion depth is calculated by the DEPTHM formula in the formulator.

Doping

$$N = \left| \frac{-2}{q\varepsilon_{S} A^{2} \frac{d(1/C^{2})}{dV}} \right|$$

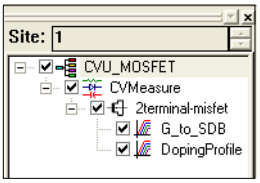
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_MOSFET** folder.
5. Open the **CVU_MOSFET.kpr** project.

The CVU_MOSFET project plan is shown in Figure 15-124.

Figure 15-124
CVU_MOSFET project plan

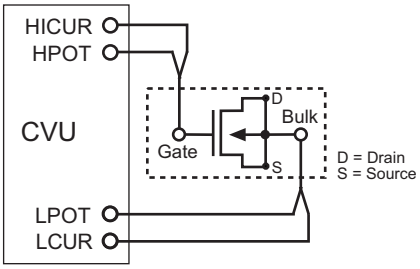


Connections

Figure 15-125 shows the basic test configuration for MOSFET testing (see Connections for details). As shown, 2-wire sense connections are used with the source, drain, and bulk terminals tied together. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-125
Two-terminal C-V test configuration for a MOSFET



Formulas and constants

Formulas and user-defined constants that are used for the application tests are summarized (in alphabetical order) in Table 15-28 (formulas) and Table 15-29 (constants). The formulas and constants are set from the formulators for the tests.

Table 15-28
Formulas for the CVU_MOSFET project

Formula Name	Formula Details	
AR (a_R)	Units: none	Description: Intermediate parameter for calculation of corrected capacitance
	AR = GP_GB-(GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*RS	
	Simplified Equation: $a_R = G - (G^2 + (2\pi fC)^2)R_S$ See Eq. 6 for details*	
CADJ (C_{ADJ})	Units: F	Description: Corrected capacitance by compensating series resistance
	CADJ = ((GP_GB^2 + (2*PI*F_GB*CP_GB)^2)*CP_GB)/(AR^2 +(2*PI*F_GB*CP_GB)^2)	
	Simplified Equation: $C_{ADJ} = \frac{(G^2 + (2\pi fC)^2)C}{a^2 + (2\pi fC)^2}$ See Eq. 4 for details*	

Table 15-28 (continued)
Formulas for the CVU_MOSFET project

CFB_CALC	Units: F	Description: Flatband capacitance
(C_{FB})	CFB_CALC = (COX_CALC*ES*AREA/(DEBYEM*1E2))/(COX_CALC+(ES*AREA/(DEBYEM*1E2)))	
	Simplified Equation: $C_{FB} = \frac{C_{OX}\left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)}{C_{OX}+\left(\frac{\varepsilon_S A}{\lambda}\right)(1xE^2)}$	
DEBYEM	Units: m	Description: Debye length (in meters)
(λ)	DEBYEM = SQRT(ES*K*TEMP/(ABS(N90W)*Q^2))*1E-2	
	Simplified Equation: $\lambda = \left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2}(1xE^{-2})$ See Eq. 10 for details	
DEPTHM	Units: m	Description: Depletion depth (in meters)
(W)	DEPTHM = 1E-2*AREA*ES*(1/COND(POX,MINPOS(CADJ),SUBARRAY(CADJ, POX, MINPOS(CADJ)),SUBARRAY(CADJ, MINPOS(CADJ),POX))-1/COX_CALC))	
	Simplified Equation: $W = A\varepsilon_S\left(\frac{1}{C}-\frac{1}{C_{OX}}\right)(1xE^{-2})$ See Eq. 2 for details*	
INVCSQR	Units: $1/F^2$	Description: Inverse square of capacitance
	INVCSQR = 1/CADJ^2	
	Simplified Equation: $INVCSQR = \frac{1}{C^2}$	
N90W	Units: none	Description: Doping density at 90% of maximum depletion depth
	N90W = AT(NDOPING, FINDD(DEPTHM,0.9*MAX(DEPTHM),2))	
NDOPING	Units: $1/cm^2$	Description: Doping density
(N)	NDOPING = ABS((-2)/(AREA^2*Q*ES)/(DELTA(COND(POX,MAXPOS(INVCSQR), SUBARRAY(INVCSQR, POX, MINPOS(CADJ)),SUBARRAY(INVCSQR, MINPOS(CADJ),POX)))/DELTA(DCV_GB)))	
	Simplified Equation: $N = \left\lvert\frac{-2}{q\varepsilon_S A^2 \frac{d(1/C^2)}{dV}}\right\rvert$ See Eq. 1 for details*	
PHIB	Units: V	Description: Bulk potential
(φ_B)	PHIB = (K*TEMP/Q*LN(ABS(N90W)/NI))*DOPETYPE	
	Simplified Equation: $\phi_B = -\frac{kT}{q}\ln\left(\frac{N_{BULK}}{N_i}\right)(DopeType)$ See Eq. 12 for details*	

Table 15-28 (continued)
Formulas for the CVU_MOSFET project

RS	Units: Ω	Description: Series resistance calculated from capacitance
(R_S)	RS = (AT(GP_GB/((2*PI*F_GB)*CP_GB),POX))^2/((1+(AT(GP_GB/ ((2*PI*F_GB)* CP_GB),POX))^2)*(AT(GP_GB,POX)))	
	Simplified Equation: $R_S = \frac{\left(\frac{G}{2\pi fC}\right)^2}{\left[1+\left(\frac{G}{2\pi fC}\right)^2\right]G}$ See Eq. 7 for details*	
TOXNM	Units: nm	Description: Calculated thickness of oxide (in nanometers)
(T_{OX})	TOXNM = (1E7*AREA*EOX)/COX_CALC	
	Simplified Equation: $T_{OX} = \frac{A\varepsilon_{OX}(1xE^7)}{C_{OX}}$ See Eq. 8 for details*	
VFB	Units: V	Description: Flatband voltage
(V_{FB})	VFB = AT(DCV_GB,POSVFB)	
	Once CFB (C_{FB}) is derived, V_{FB} is interpolated from the closest V_{GS} values.	

* Details for referenced equations are found in the documentation for Key concepts for MOScap testing.

Table 15-29
Constants for the CVU_MOSFET project

Constant	Default Value	Units	Description
AREA	10.404E-03	cm^2	Gate area of device
DOPETYPE	1E+00	none	1 = P-type, -1 = N-type
EOX	340.0E-15	F/cm	ε_{OX} - Permittivity of oxide
ES	1.03E-12	F/cm	ε_S - Semiconductor permittivity
Q	160.218E-21	C	Charge on an electron
TEMP	293E+00	K	Test temperature

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

G_to_SDB and DopingProfile ITMs

Test summaries

Both these Interactive Test Modules (ITMs) perform measurements at each step of a linear voltage sweep, but the graphs that they generate differ:

- The G_to_SDB test generates a Capacitance versus Voltage graph.
- The DopingProfile test generates a Doping Concentration versus Depletion Depth graph.

Parameter settings

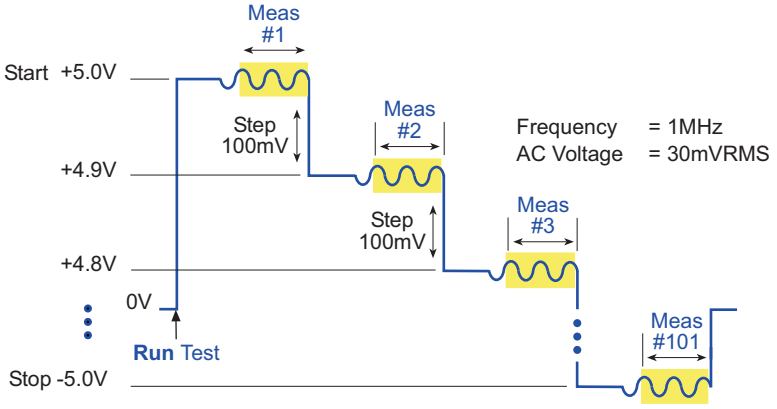
The default parameter settings for these tests are listed in Table 15-30, and the voltage sweep used for these tests is shown in Figure 15-126.

Table 15-30
Parameter settings (G_to_SDB and DopingProfile ITMs)

Force (Figure 15-128)	**Measure** (Figure 15-128)	**Timing** (Figure 15-127)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 5 V Start: 5 V Stop: -5 V Step: -0.1 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 30 mV Data Points: 101	Cp-Gp DCV Frequency I Range: 30µA [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 0 s Timestamp: Enabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-128)	**CVU Compensation** [3] (Figure 15-128)	
Gate: DC Source V Gate: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-126
Linear voltage sweep (G_to_SDB and DopingProfile ITM)



In the Project Navigator, double-click an ITM to display the Definition tab, which is shown in Figure 15-127.

Figure 15-127
Definition tab (G_to_SDB and DopingProfile ITM)

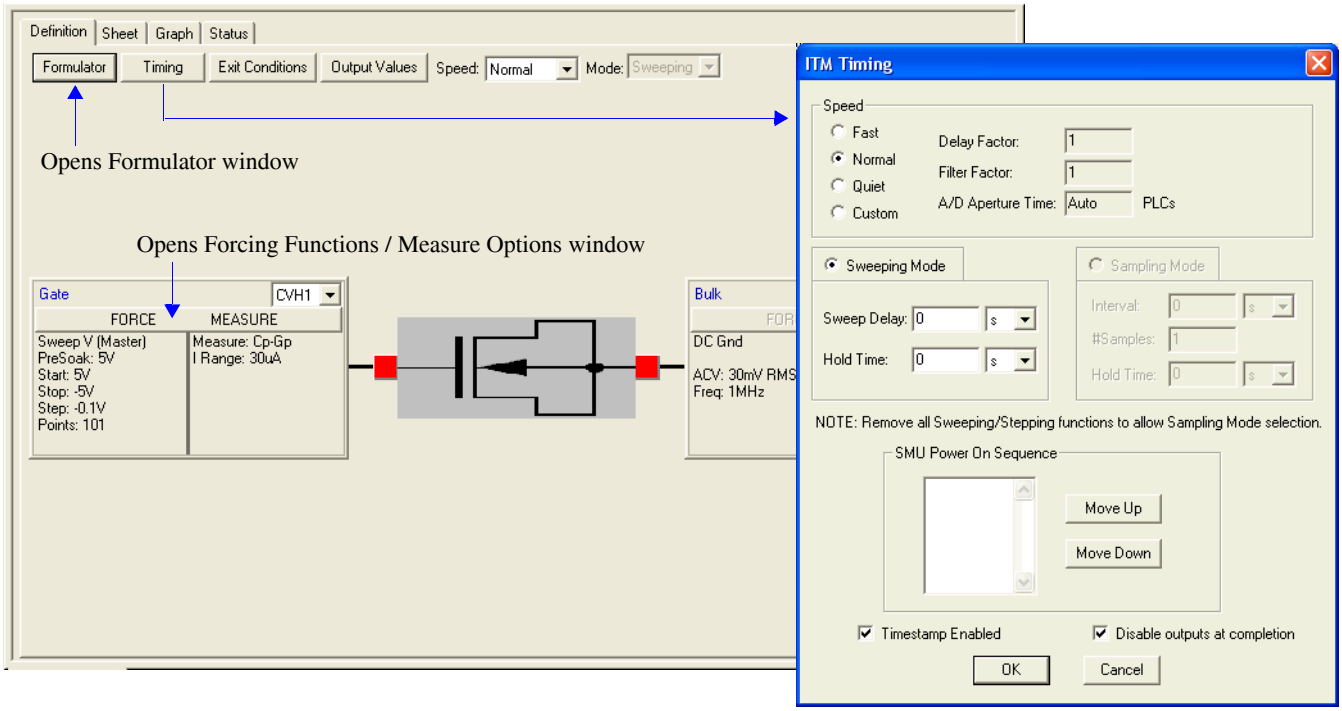
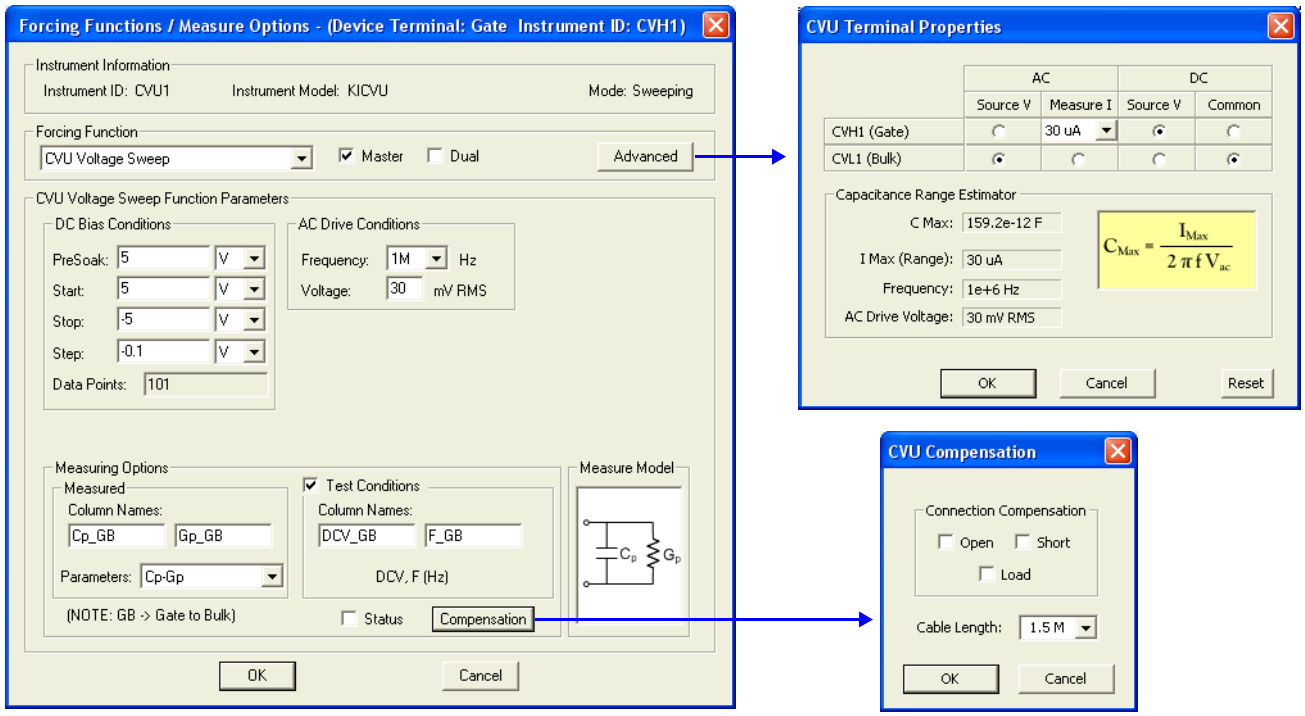


Figure 15-128
Forcing Functions / Measure Options window (G_to_SDB and DopingProfile ITMs)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Cp_GB Measured parallel capacitance.
- Gp_GB Measured conductance.
- DCV_GB Forced DC bias voltage.
- F_GB Forced test frequency.
- Formulas Formulator calculation results.

Note: GB = gate-to-bulk.

Graphs

The graphs are displayed in the Graph tabs (see Figure 15-129 and Figure 15-130). The C-V graph includes the results for the COX_CALC, CFB_CALC, VFB, and N90W formula calculations.

Figure 15-129
Graph tab (G_to_SDB ITM)

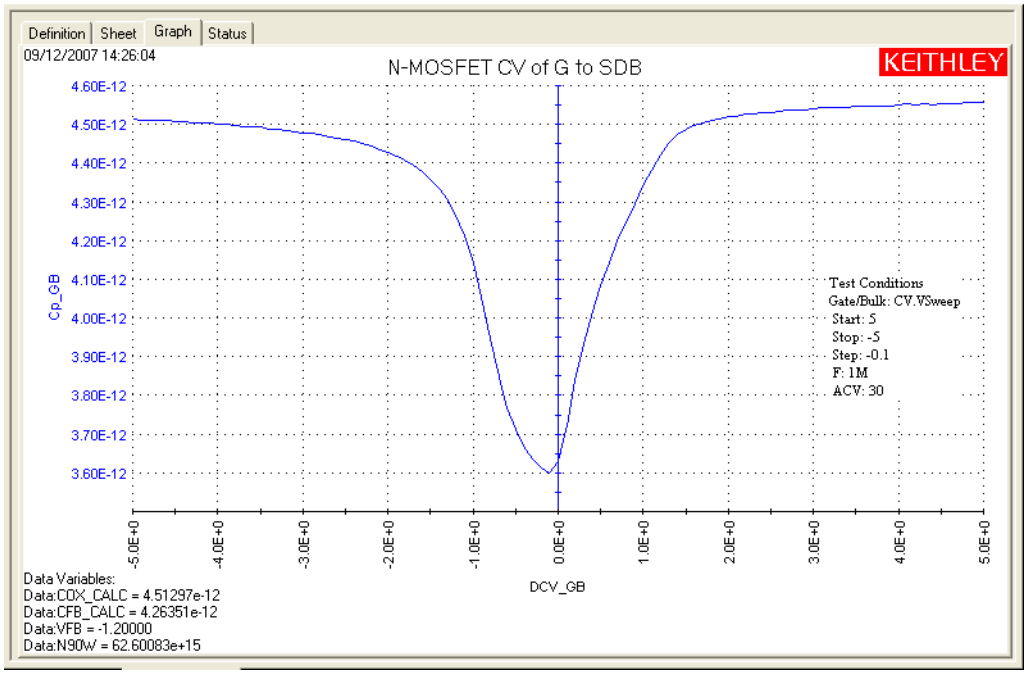
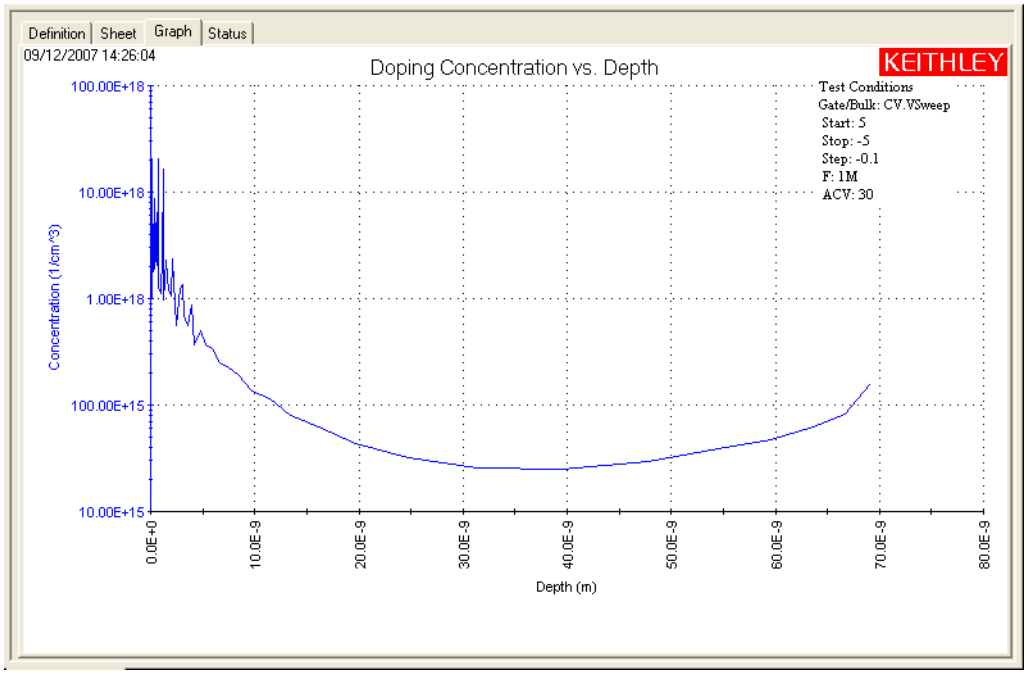


Figure 15-130
Graph tab (DopingProfile ITM)



CVU_nanowire

Project Plan

Key concepts

C-V measurements on semiconductor nanowires and nanowire-based devices can be used to derive important characteristics about the device, including mobility, carrier density, and device speed. Sometimes the capacitance is plotted as a function of channel length or gate length. These capacitance measurements can often be quite small, <1pF. As a result, using proper techniques to reduce parasitic capacitance from affecting measurement accuracy is important.

Project summary

The two Interactive Test Modules (ITMs) in this project perform C-V sweeps on a two-terminal nanowire device. Both tests are similar, but use different drive frequencies. Capacitance versus voltage graphs are generated by the tests.

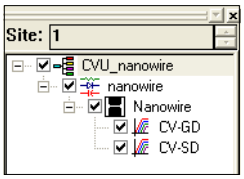
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_nanowire** folder.
5. Open the **CVU_nanowire.kpr** project.

The CVU_nanowire project plan is shown in Figure 15-131.

Figure 15-131
CVU_nanowire project plan



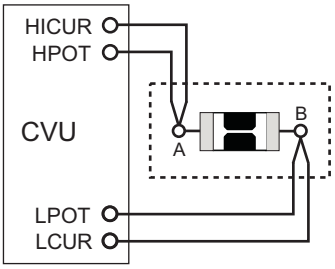
Connections

Figure 15-132 shows the basic test configuration; Connections provides details. The CV-GD ITM is used to test a nanowire on a wafer. The CV-SD ITM can be used to test a nanowire on a wafer, or a discrete nanowire device.

Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-132
Basic configuration to test a nanowire device



Formulas and constants

This project uses two formulas (no constants), which are shown in Table 15-31.

Table 15-31
Formula for the CVU_nanowire

Formula Name	Formula Details	
CAVG	Units: F	Description: Calculates the average capacitance.
	CAVG = AVG(CP_AB)	
STD_DEV	Units: none	Description: Calculates the standard deviation of the capacitance measurements.
	STD_DEV = STDEV(CP_AB)	

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

CV-GD and CV-SD ITMs

Test summaries

CV-GD ITM — This ITM performs a C-V sweep (gate-to-drain, 1 MHz) across a nanowire that is connected to the backside of a wafer. This test generates a C versus V graph.

CV-SD ITM — This ITM performs a C-V sweep (source-to-drain, 100 kHz) across a nanowire. This test generates a C versus V graph. It also calculates average capacitance and standard deviation.

NOTE *Both ITMs are configured the same except for the drive frequency, as indicated above.*

Parameter settings

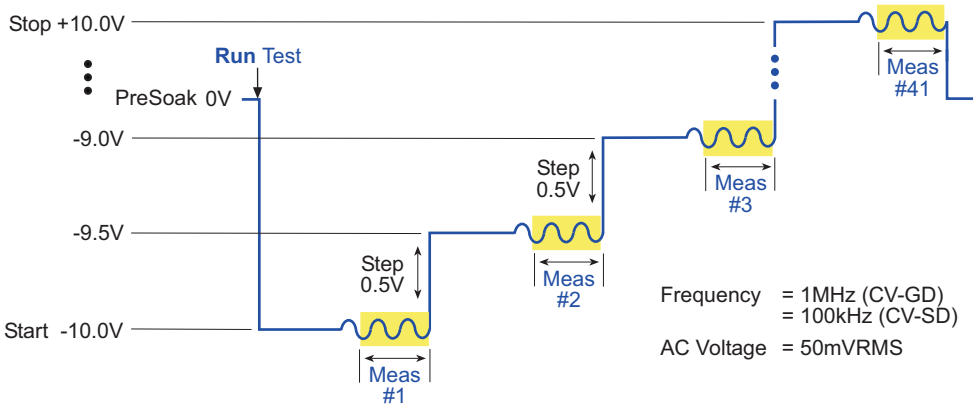
The default parameter settings for this test are listed in Table 15-32. The voltage sweep used for this test is shown in Figure 15-133.

Table 15-32
Parameter settings (CV-GD and CV-SD ITMs)

Force (Figure 15-135)	**Measure** (Figure 15-135)	**Timing** (Figure 15-134)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -10 V Stop:10 V Step: 0.5 V AC Drive Conditions Frequency: 1 MHz (CV-GD) 100 kHz (CV-SD) AC Voltage: 50 mV Data Points: 41	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Quiet Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-135)	**CVU Compensation** [3] (Figure 15-135)	
Terminal A: DC Source V Terminal A: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-133
Linear voltage sweep (CV-GD and CV-SD ITMs)



In the Project Navigator, double-click the **CV-GD** or **CV-SD** ITM to display the Definition tab. Figure 15-134 shows the Definition tab for the CV-GD test.

Figure 15-134
Definition tab (CV-GD ITM)

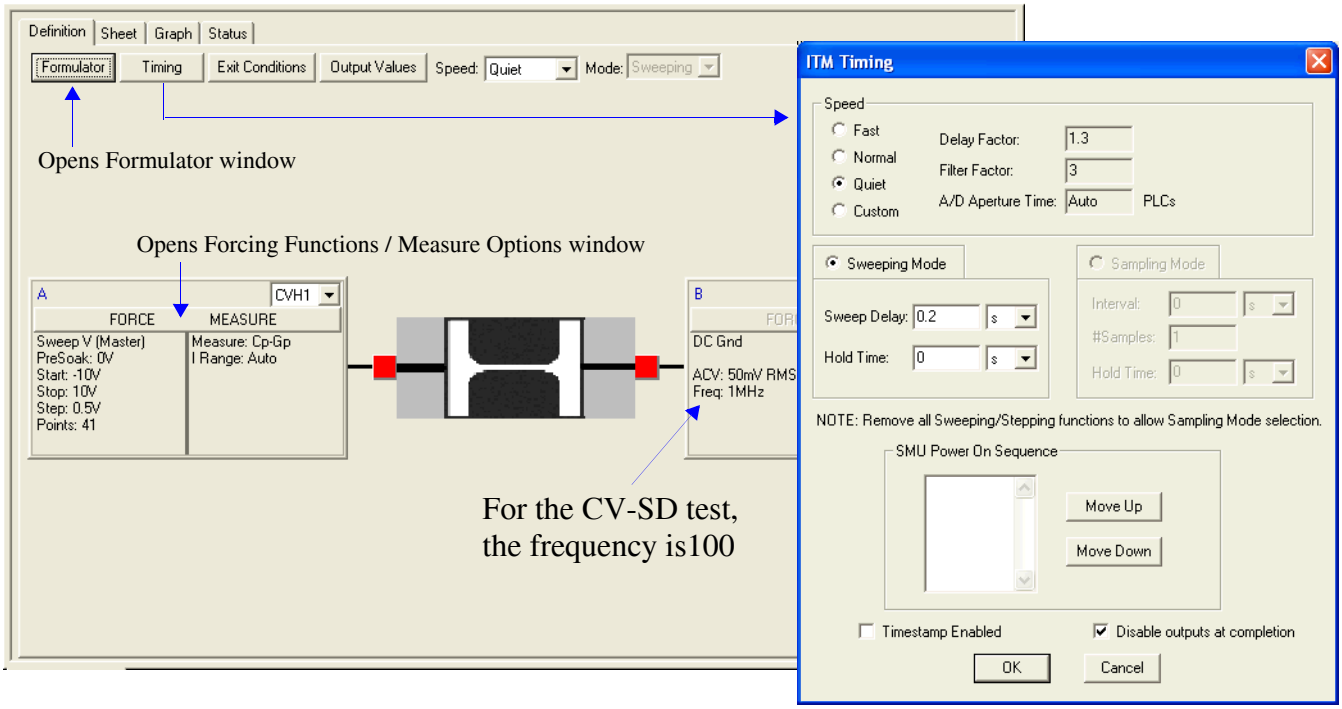
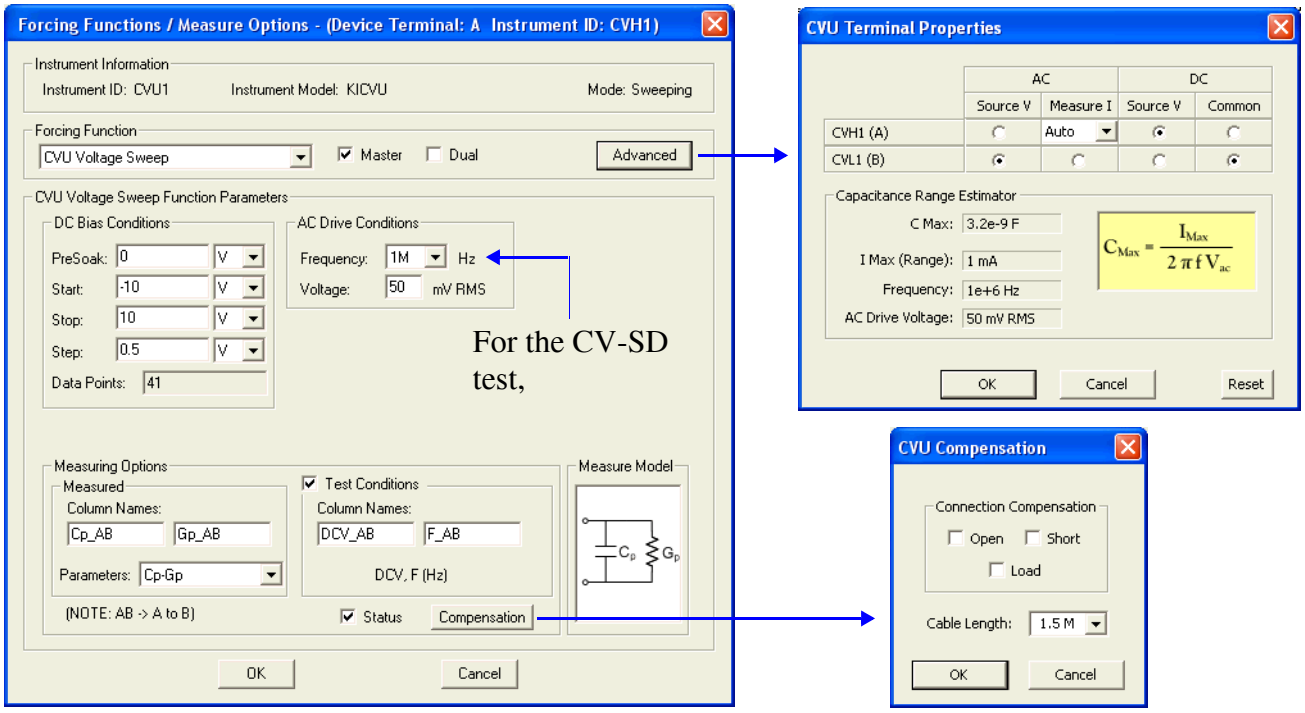


Figure 15-135
Forcing Functions / Measure Options window (CV-GD ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Cp_AB Measured parallel capacitance.
- Gp_AB Measured conductance.
- DCV_AB Forced DC bias voltage.
- F_AB Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

  Note: AB = Terminal A to Terminal B.

Graph

The graphs are displayed in the Graph tabs (see Figure 15-136 and Figure 15-137). The graph for CV-SD includes the calculation results average capacitance and standard deviation.

Figure 15-136
Graph tab (CV-GD ITM)

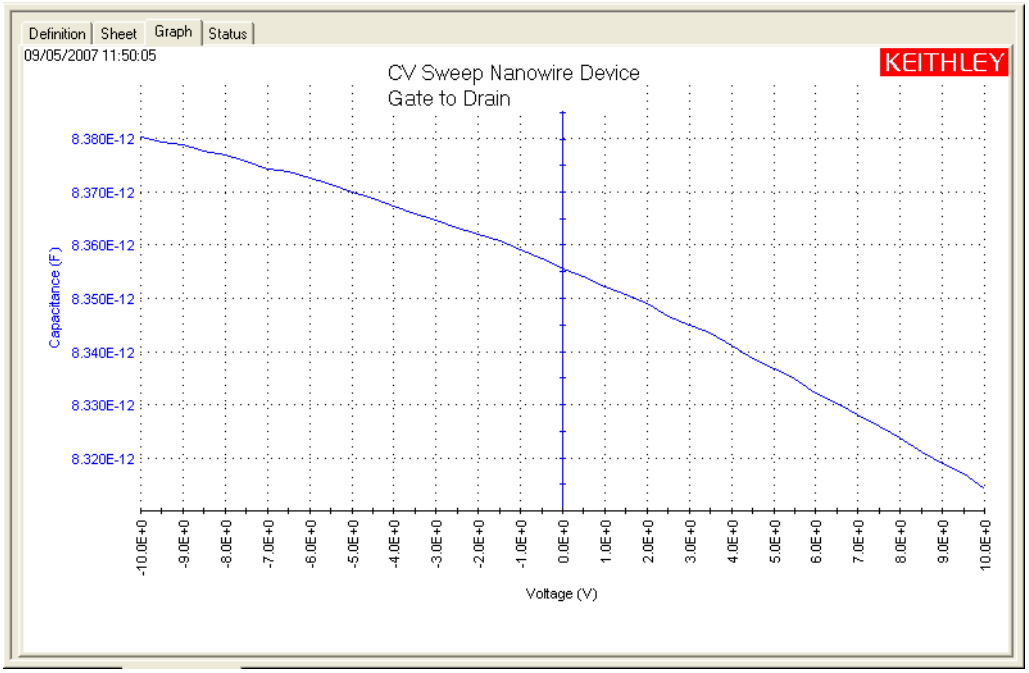
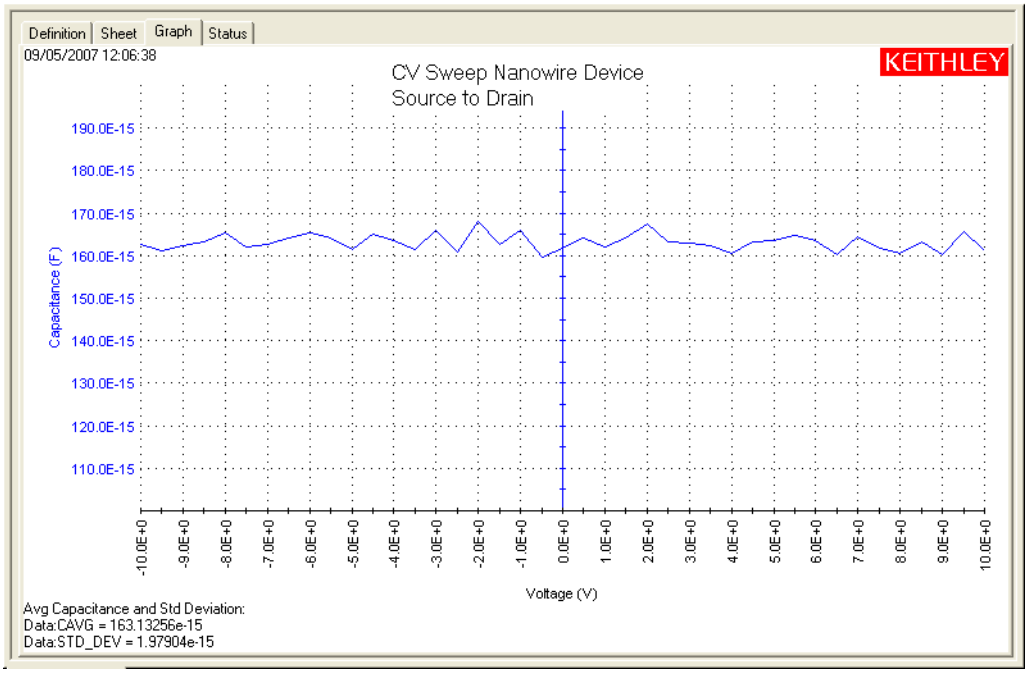


Figure 15-137
Graph tab (CV-SD ITM)



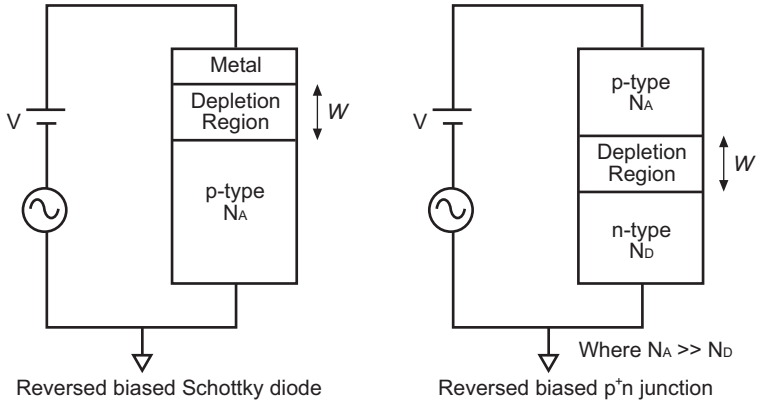
CVU_PNjunction

Project Plan

Key concepts for PN junction and Schottky diode testing

C-V measurements on semiconductor junctions are based on the fact that the width (*W*) of the depletion region of the junction is not constant, but varies with the DC voltage across the junction. This capacitance, related to the space charge region of the semiconductor junction, is known as the junction capacitance. This concept is illustrated in Figure 15-138, showing a reverse biased Schottky diode and a reverse biased PN junction.

Figure 15-138
Semiconductor junctions



Since the junction can be modeled after a parallel plate capacitor, the junction capacitance is calculated as follows:

$$C = \frac{\varepsilon_S A}{W} \qquad \textit{Eq. 1}$$

Where: C = Junction capacitance (F)
ε_S = Semiconductor permittivity (1.034 E-12 F/cm for silicon)
A = Area of junction (cm^2)
W = Depletion width (cm)

However, unlike the parallel plate capacitor, the depletion layer width (*W*) is not a constant, but is dependent on the applied voltage. From the previous equation the depletion depth, *W*, can be calculated as follows:

$$W = \frac{\varepsilon_S A}{C} \qquad \textit{Eq. 2}$$

From the measured capacitance and the voltage, the doping density can be calculated as follows:

$$N(W) = \frac{2}{q\varepsilon_S A^2\left[\frac{d(1/C^2)}{dV}\right]} \qquad \textit{Eq. 3}$$

Where: N(W) = Doping density (cm^{-3})
A = Area of junction (cm^2)
C = Junction capacitance (F)
ε_S = Semiconductor permittivity (1.034 E-12 F/cm for silicon)
q = Electron charge (1.60219 x 10^{-19} C)
V = Junction voltage

Project plan tests

The project plan (CVU_PNjunction) for this application is used to measure the capacitance of a pn junction or Schottky diode as a function of the DC bias voltage across the device. The PN junctions must be highly asymmetrical p^+n or n^+p junctions, which means that one side of the junction is much more highly doped than the other side. If this is the case, then the effects of the space charge region spreading into the more highly doped area can be ignored. The project consists of three ITMs: CVsweep, C-2vsV, and DopingProfile.

CVsweep ITM

The junction capacitance is measured as a function of the DC bias voltage while the junction is reverse biased. The AC bias is typically in the 10 mV to 20 mV range.

C-2vsV ITM

Instead of plotting dC/dV, it is sometimes desirable to view the data as a $1/C^2$ versus V curve. The doping density (*N*) can be derived from the slope of this curve because *N* is linearly related to the capacitance (see Eq. 3 above).

The built-in potential (φ_B) of the diode can be derived from the intersection of the $1/C^2$ curve and the horizontal axis. The graph option, linear Line Fits, can be used to easily derive both the doping density (*N*) and the built-in voltage on the x-axis.

DopingProfile ITM

Because any change in the junction voltage yields a capacitance and thus a change in width of the space charge region of the junction, CV measurements can be used for doping profiling. At each voltage step, the doping density (*N*) and the corresponding depth (*W*) can be calculated from Eq. 2 and Eq. 3 above.

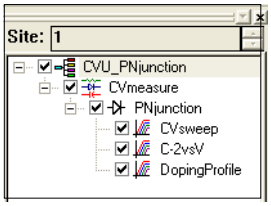
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_PNjunction** folder.
5. Open the **CVU_PNjunction.kpr** project.

The CVU_PNjunction project plan is shown in Figure 15-139.

Figure 15-139
CVU_PNjunction project plan

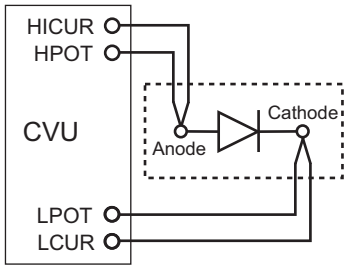


Connections

Figure 15-140 shows the basic test configuration; Figure 15-11 shows the actual connections to a semiconductor wafer. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-140
Basic configuration to test a semiconductor junction



Formulas and constants

Formulas and user-defined constants that are used for the tests are listed in Table 15-33 (formulas) and Table 15-34 (constants). The formulas and constants are set from the formulators for the tests.

Table 15-33
Formulas for the CVU_PNjunction project

Formula Name	Formula Details	
DEPTH (*W*)	Units: cm	Description: Width of depletion (junction) region
	DEPTH = (ES*AREA)/(CP_AC)	
	Simplified Equation: $W = \frac{E_S A}{C}$ See Eq. 1 for details*	
INV_C (*C*)	Units: 1/F	Description: Inverse capacitance
	INV_C = 1/(CP_AC)^2	
	Simplified Equation: $C = \frac{1}{C^2}$	
DOPING (*N*)	Units: 1/cm^3	Description: Majority carrier concentration
	DOPING = ABS(2/Q/ES/AREA^2/DIFF(INV_C, DCV_AC))	
	Simplified Equation: $N = \left\| \frac{2}{q\varepsilon_S A^2 \frac{d(1/C^2)}{dV}} \right\|$	

* Details for referenced equations are found in the documentation for Key concepts for PN junction and Schottky diode testing.

Table 15-34
Constants for the CVU_PNjunction project

Constant	Default Value	Units	Description
AREA	100.0E-06	cm^2	Gate area of device
ES	1.034E-12	F/cm	ε_S - Semiconductor permittivity

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

CVsweep ITM

Test summary

For this ITM, the PN junction capacitance is measured as a function of the DC bias voltage while; the junction is reverse biased. This test performs a capacitance measurement at each step of a user-configured linear voltage sweep. From the acquired C-V data, parameters are calculated using the formulator. See CVsweep ITM for more information about this test.

Parameter settings

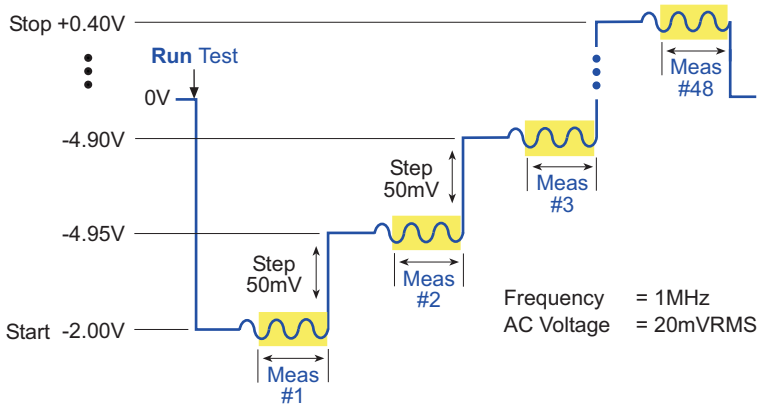
The default parameter settings for this test are listed in Table 15-35, and the voltage sweep used for this test is shown in Figure 15-141.

Table 15-35
Parameter settings (CVsweep ITM)

Force (Figure 15-143)	**Measure** (Figure 15-143)	**Timing** (Figure 15-142)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -2 V Stop: 0.4 V Step: 0.05 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 20 mV Data Points: 48	Cp-Gp DCV Frequency I Range: Auto [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-143)	**CVU Compensation** [3] (Figure 15-143)	
Anode: DC Source V Cathode: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-141
Linear voltage sweep (CVsweep ITM)



In the Project Navigator, double-click the **CVsweep** ITM to display the Definition tab, which is shown in Figure 15-142.

Figure 15-142
Definition tab (CVsweep ITM)

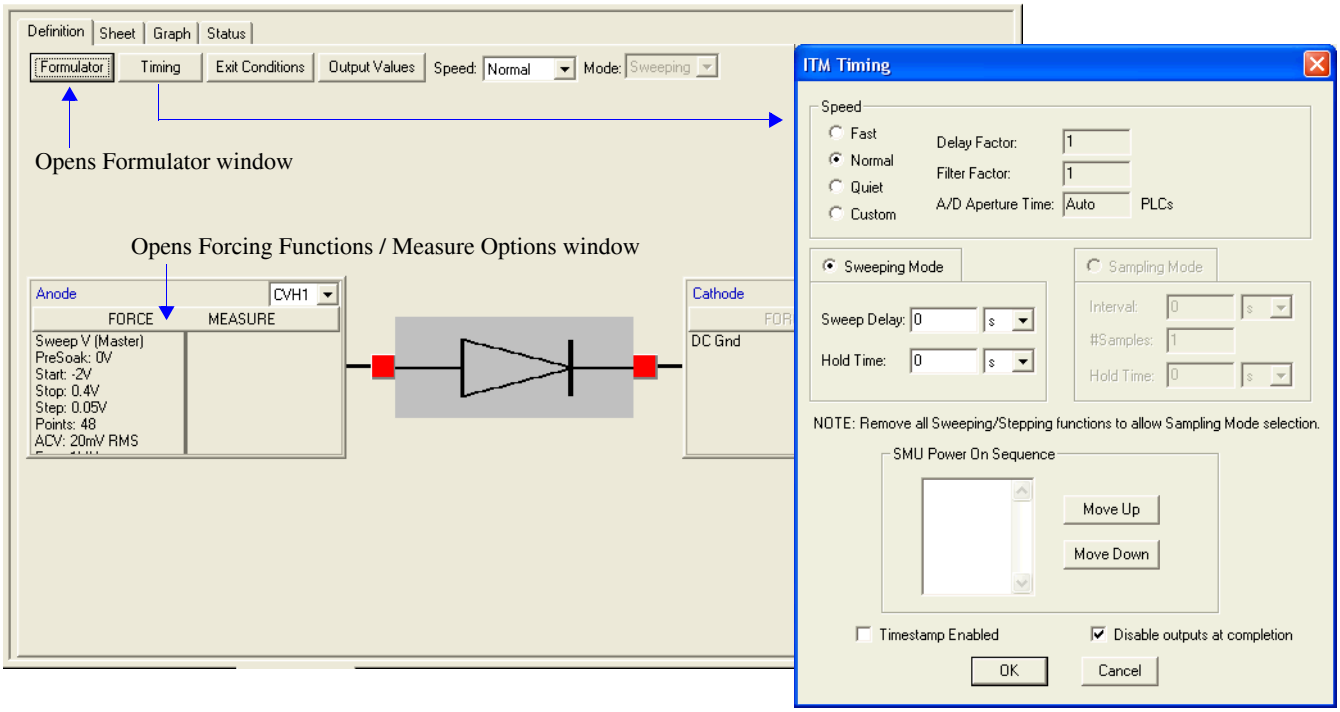
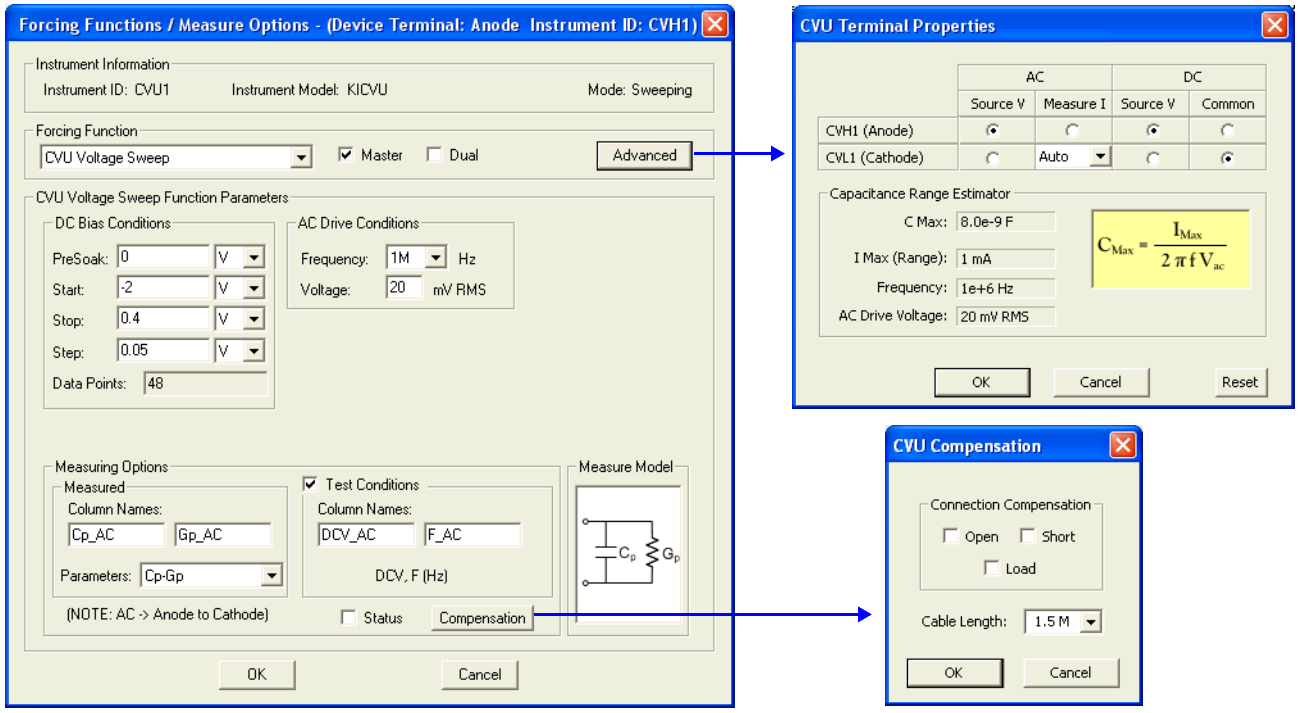


Figure 15-143
Forcing Functions / Measure Options window (CVsweep ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

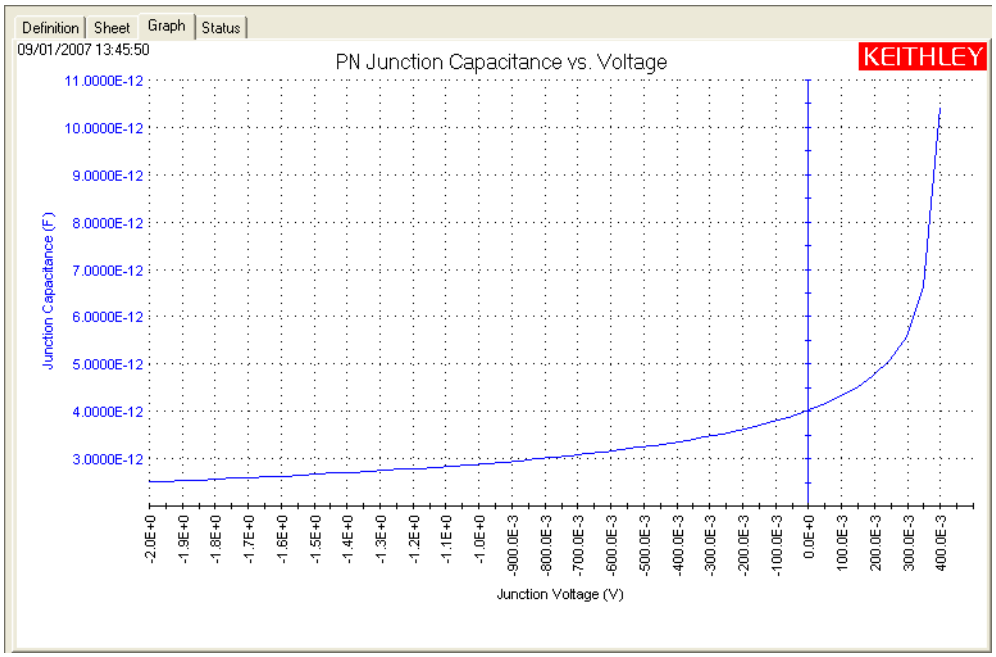
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- Formulas Formulator calculation results.

Note: AC = anode-to-cathode.

Graph

The graph is displayed in the Graph tab (see Figure 15-144).

Figure 15-144
Graph tab (CVsweep ITM)



C-2vsV ITM

Test summary

Instead of plotting dC/dV, it is sometimes desirable to view the data as a $1/C^2$ versus V curve. The graph option, linear Line Fits, can be used to derive both the doping density (*N*) and the built-in voltage on the x-axis. See C-2vsV ITM for more information about this test.

Parameter settings

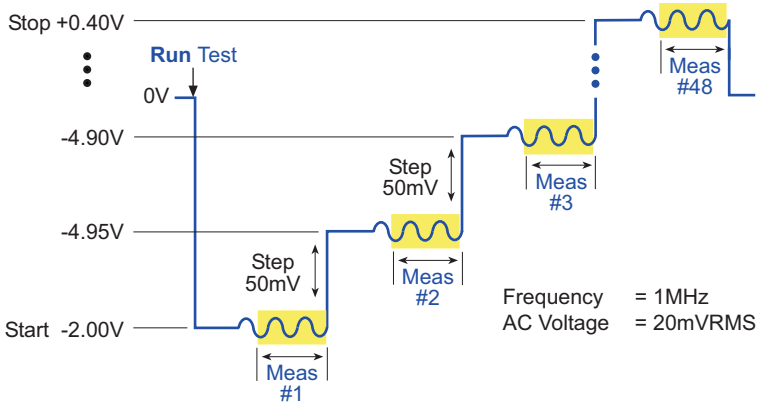
The default parameter settings for this test are listed in Table 15-36. The voltage sweep used for this test is shown in Figure 15-145.

Table 15-36
Parameter settings (C-2vsV ITM)

Force (Figure 15-147)	**Measure** (Figure 15-147)	**Timing** (Figure 15-146)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -2 V Stop: 0.4 V Step: 0.05 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 20 mV Data Points: 48	Cp-Gp DCV Frequency I Range: Auto [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-147)	**CVU Compensation** [3] (Figure 15-147)	
Anode: DC Source V Cathode: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-145
Linear voltage sweep (C-2vsV ITM)



In the Project Navigator, double-click the **C-2vsV** ITM to display the Definition tab, which is shown in Figure 15-146.

Figure 15-146
Definition tab (C-2vsV ITM)

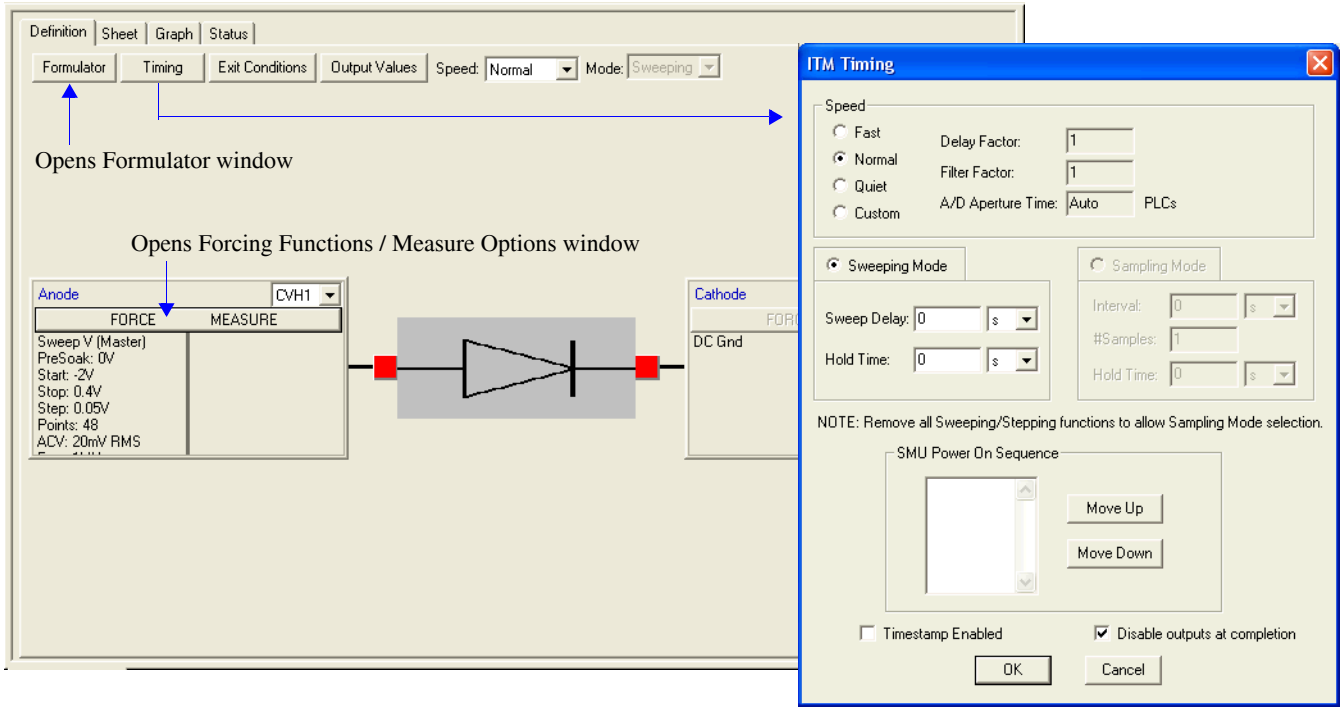
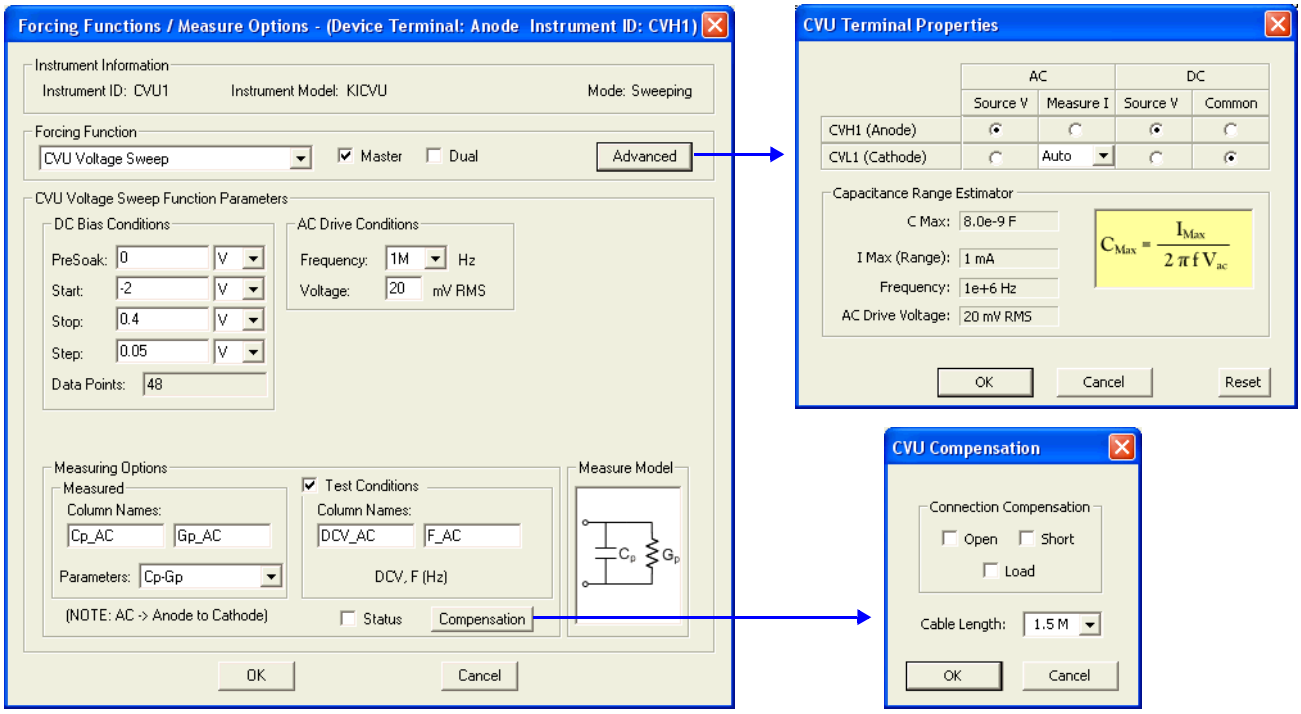


Figure 15-147
Forcing Functions / Measure Options window (C-2vsV ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

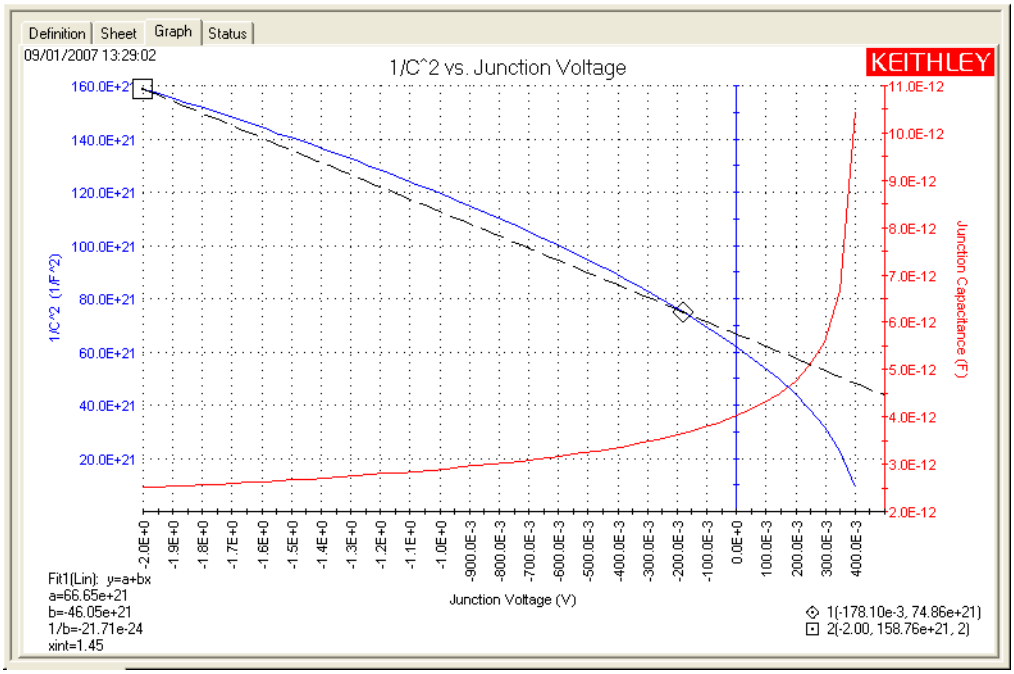
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- Formulas Formulator calculation results.

Note: AC = anode-to-cathode.

Graph

The graph is displayed in the Graph tab (see Figure 15-148).

Figure 15-148
Graph tab (C-2vsV ITM)



DopingProfile ITM

Test summary

This ITM performs C-V measurements which are then used for doping profiling. A graph is generated to show doping density versus depth. See DopingProfile ITM for more information about this test.

Parameter settings

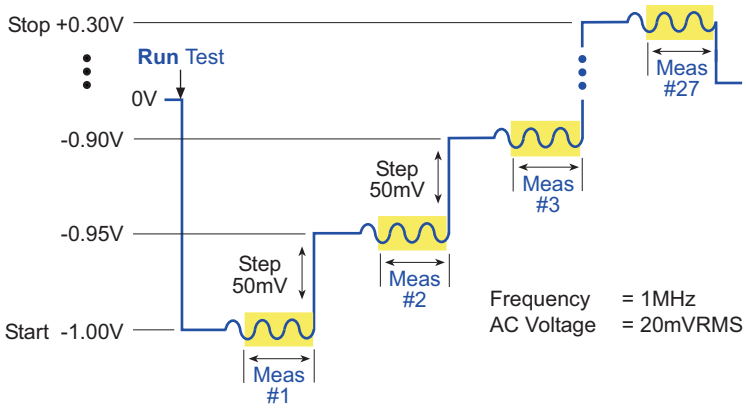
The default parameter settings for this test are listed in Table 15-37. The voltage sweep used for this test is shown in Figure 15-149.

Table 15-37
Parameter settings (DopingProfile ITM)

Force (Figure 15-151)	**Measure** (Figure 15-151)	**Timing** (Figure 15-150)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -1 V Stop: 0.3 V Step: 0.05 V AC Drive Conditions Frequency: 1 MHz AC Voltage: 20 mV Data Points: 27	Cp-Gp DCV Frequency I Range: Auto [1] Status: Disabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.1 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-151)	**CVU Compensation** [3] (Figure 15-151)	
Anode: DC Source V Cathode: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-149
Linear voltage sweep (DopingProfile ITM)



In the Project Navigator, double-click the **DopingProfile** ITM to display the Definition tab, which is shown in Figure 15-150.

Figure 15-150
Definition tab (DopingProfile ITM)

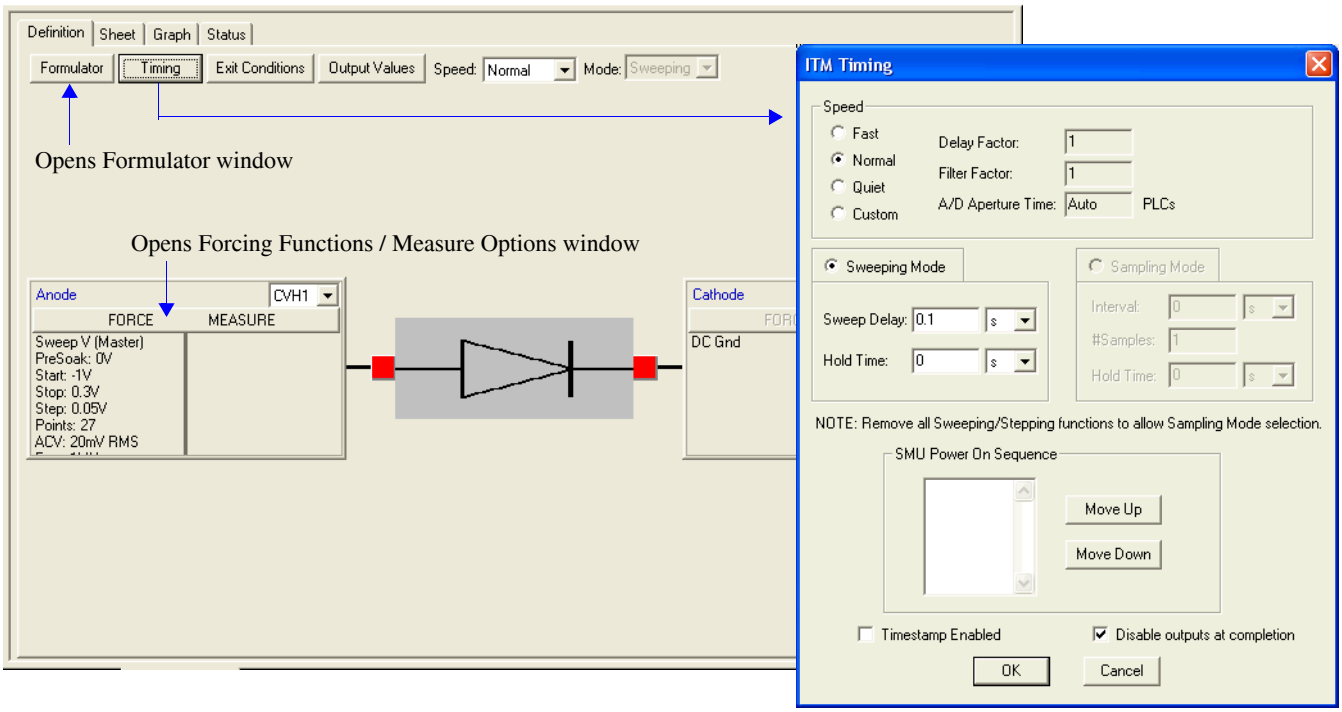
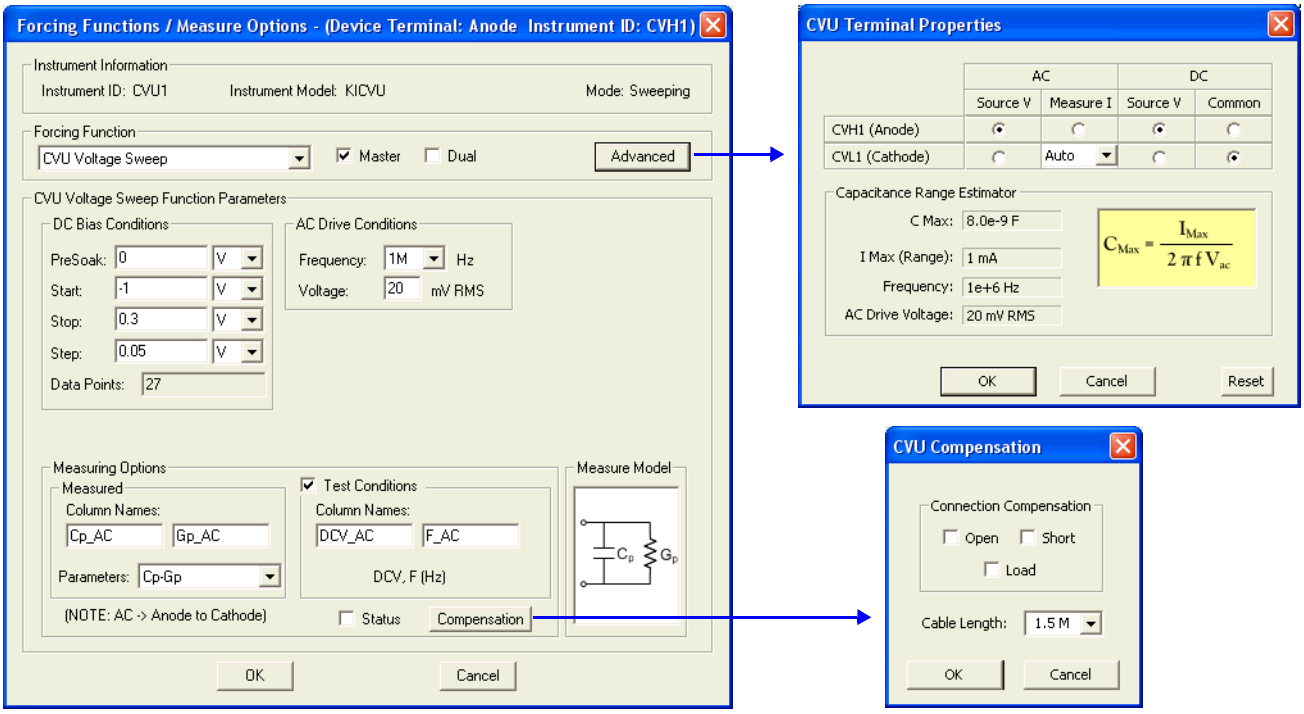


Figure 15-151
Forcing Functions / Measure Options window (DopingProfile ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

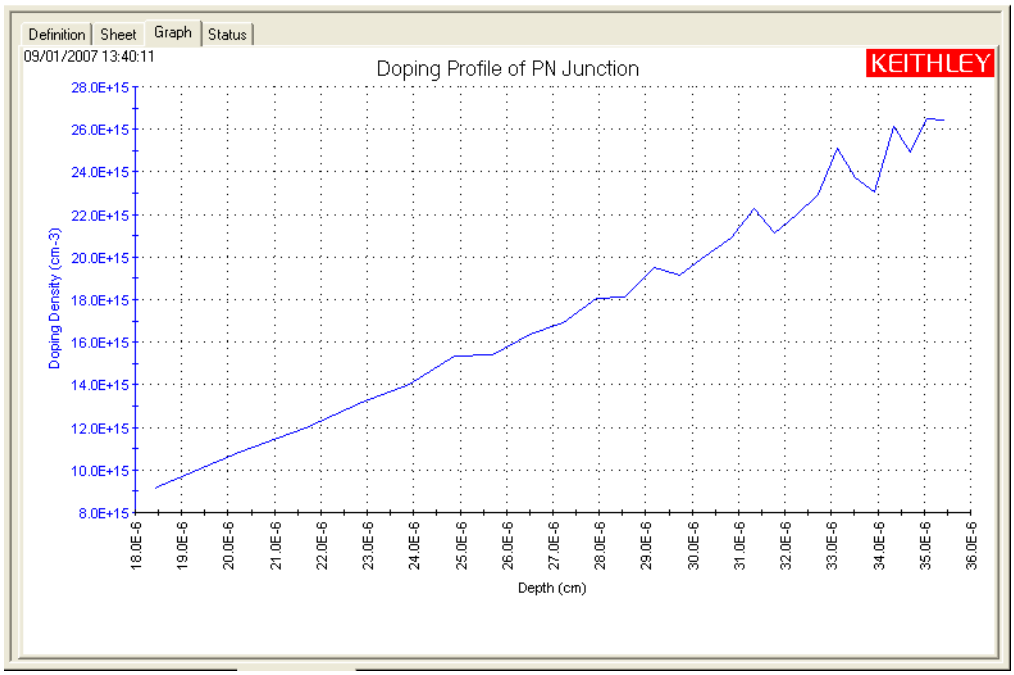
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- Formulas Formulator calculation results.

Note: AC = anode-to-cathode.

Graph

The graph is displayed in the Graph tab (see Figure 15-152).

Figure 15-152
Graph tab (DopingProfile ITM)



CVU_PVcell

Project Plan

Key concepts

To determine the electrical characteristics of a photo voltaic (PV) cell, a Model 4200-SMU and Model 4200-CVU are used to perform I-V, C-V, and C-F sweeps. Not only are graphs (I versus V, C versus V, and C versus F) generated from the collected current, capacitance, frequency and voltage data, but formulator calculations (see Table 15-38) are performed to determine the following electrical characteristics of the PV cell:

I-V testing: The formulator is used to calculate power (P), current (CURR) maximum power (PMAX), maximum current (IMAX), maximum voltage (VMAX), short-circuit current (ISC), open-circuit voltage (VOC), and fill factor (FF).

C-V testing: The formulator is used to calculate inverse capacitance (INV_C) and the doping density (N). A graph ($1/C^2$ versus V) is generated using the calculated inverse capacitance values.

The testing process is typically repeated using different light intensities and temperature conditions.

NOTE *Details on PV cell testing are provided by Application Note #2876, and is available from Keithley Instruments.*

Project summary

Project plan tests are divided into two parts to test PV cells: I-V testing, and C-V testing:

I-V testing:

- Generates an I versus V graph for a forward-biased PV cell. formulator formulas calculate the electrical characteristics of the cell.
- Generates an I versus V graph for a reversed-biased PV cell.

C-V testing:

- Generates a C versus V graph for a reversed-biased PV cell.
- Generates a $1/C^2$ and C versus V graph for a PV cell. $1/C^2$ and doping density are calculated by the formulator.
- Generates a C versus F graph for a PV cell.

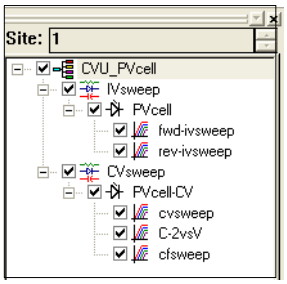
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **_CV** folder.
4. Double-click the **CVU_PVcell** folder.
5. Open the **CVU_PVcell.kpr** project.

The CVU_PVcell project plan is shown in Figure 15-153.

Figure 15-153
CVU_PVcell project plan



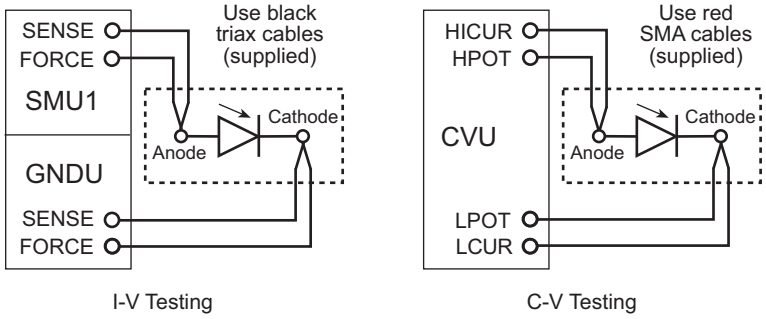
Connections

Figure 15-154 shows the basic test configurations for this project:

- **I-V Testing**: The basic configuration shows four-wire sensing using a Model 4200-SMU or Model 4210-SMU. Four-wire sensing is needed to achieve optimum performance. See Section 4 for details on Model 4200-SMU connections.
- **C-V Testing**: See Connections for details on Model 4200-CVU connections. Use only the supplied (red) 100 Ω SMA cables for connections to the Model 4200-CVU. Be sure that all used SMA cables are the same length (1.5 m or 3 m).

NOTE *After making connections (and anytime the connection setup is changed), be sure to use the Confidence Check diagnostic tool and perform connection compensation tests (see Confidence Check and Connection compensation for details).*

Figure 15-154
Basic configurations to test PV cells



Formulas and constants

Formulas and user-defined constants that are used for the tests are listed (in alphabetical order) in Table 15-38 (formulas) and Table 15-39 (constants). The formulas and constants are set from the formulators for the tests.

Table 15-38
Formulas for the CVU_PVcell project

Formula Name	Formula Details	
CURR	Units: A	Description: PV cell current
	CURR = ABS(ANODEI)	
	Absolute value of current output.	
FF	Units: none	Description: Fill factor
	FF = (IMAX*VMAX)/(ISC*VOC)	
	Simplified Equation: $FF = \frac{I_{MAX}V_{MAX}}{I_{SC}V_{OC}}$ I_{MAX} = IMAX I_{SC} = ISC V_{MAX} = VMAX V_{OC} = VOC	
IMAX	Units: A	Description: Cell current where the power output of the cell is greatest (PMAX)
	IMAX = ABS(AT(ANODEI, MAXPOS(P)))	
ISC (I_{SC})	Units: A	Description: Short circuit current
	ISC = ABS(AT(ANODEI, FINDD(ANODEV, 0, FIRSTPOS(ANODEV))))	
	I_{SC} is the point in I-V data where V = 0.	
N ($N_{(a)}$)	Units: 1/cm^3	Description: Doping density
	N = ABS(2/Q/ES/AREA^2/DIFF(INV_C2, DCV_AC))	
	Simplified Equation: $N_{(a)} = \left\| \frac{2}{q\varepsilon_S A^2 \frac{d(1/C^2)}{dV}} \right\|$ $N_{(a)}$ = Doping density; q = Electron charge (C); E_S = Semiconductor permittivity for silicon (F/cm); A = Area (cm^2); C = Measured capacitance (F); V = Applied voltage (V)	
P	Units: W	Description: Power
	P = CURR*ANODEV	
	Simplified Equation: $P = I \times V$ I = cell current; V = cell voltage	
PMAX (P_{MAX})	Units: W	Description: Maximum power point
	PMAX = MAX(P)	
	Simplified Equation: $P_{MAX} = I_{MAX} \times V_{MAX}$	
VMAX	Units: V	Description: Cell voltage where the power output of the cell is greatest (PMAX)
	VMAX = AT(ANODEV, MAXPOS(P))	
VOC (V_{OC})	Units: V	Description: Open circuit voltage
	VOC = AT(ANODEV, FINDU(ANODEI, 0, LASTPOS(ANODEI)))	
	V_{OC} is the point in I-V data where I = 0	

Table 15-38 (continued)
Formulas for the CVU_PVcell project

INV_C2	Units: $1/F^2$	Description: Inverse capacitance
(*C*)	INV_C2 = 1/CP_AC^2	
	Simplified Equation: $C = \frac{1}{C^2}$	

Table 15-39
Constants for the CVU_PVcell project

Constant	Default Value	Units	Description
AREA	8E+00	cm^2	Gate area of device
ES	1.034E-12	F/cm	ε_S - Semiconductor permittivity

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

fwd-ivsweep ITM

Test summary

NOTE *Make sure the PV cell is connected to the Model 4200-SMU (see Connections).*

This ITM uses a Model 4200-SMU to perform a forward-biased voltage sweep on a PV cell. Current is measured on each step of the sweep. An I versus V graph is generated from the collected data. It also performs the formulator calculations to determine the electrical characteristics of the cell.

When the test is started, the SMU sweeps from -50 mV to 250 mV in 2mV steps. A total of 151 current measurements are performed.

Force, measure, and timing settings

In the Project Navigator, double-click the **fwd-ivsweep** ITM to display the Definition tab. The force-measure and timing settings for the Model 4200-SMU are shown in Figure 15-155 and Figure 15-156. The **GNDU** is selected from the Definition tab.

NOTE *Details on KITE configuration for SMUs are provided in Section 6.*

Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- Anodel Measured current.
- AnodeV Forced voltage.
- Formulas Formulator calculation results.

Graph

The graph is displayed in the Graph tab (see Figure 15-157). It includes the calculation results for PMAX, ISC, and VOC.

Figure 15-155
Definition tab (fwd-ivsweep ITM)

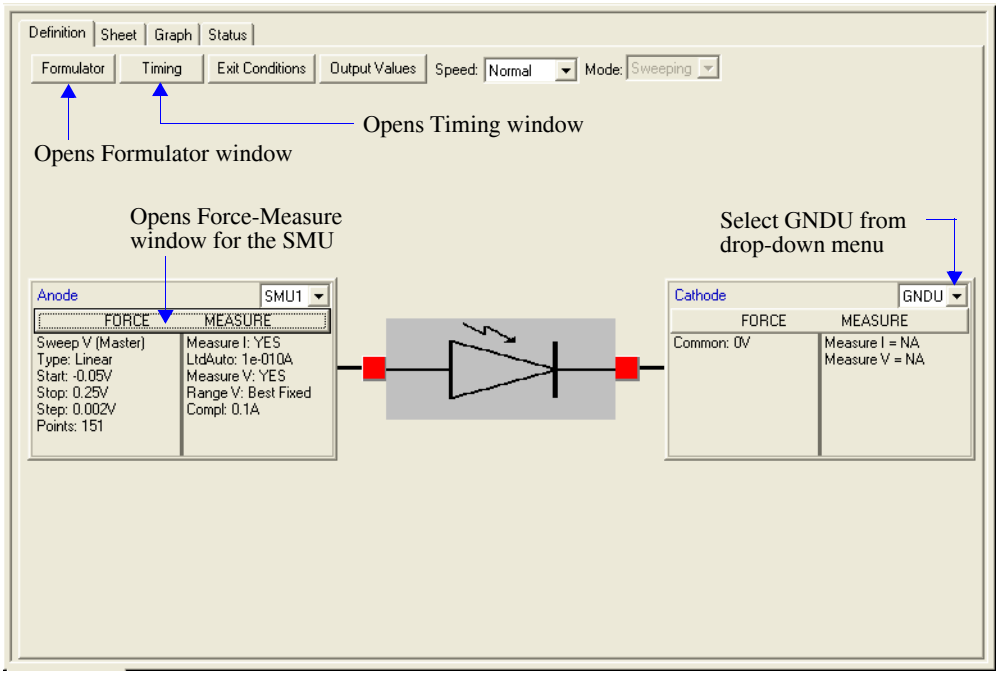


Figure 15-156
Forcing Functions / Measure Options and Timing windows (fwd-ivsweep ITM)

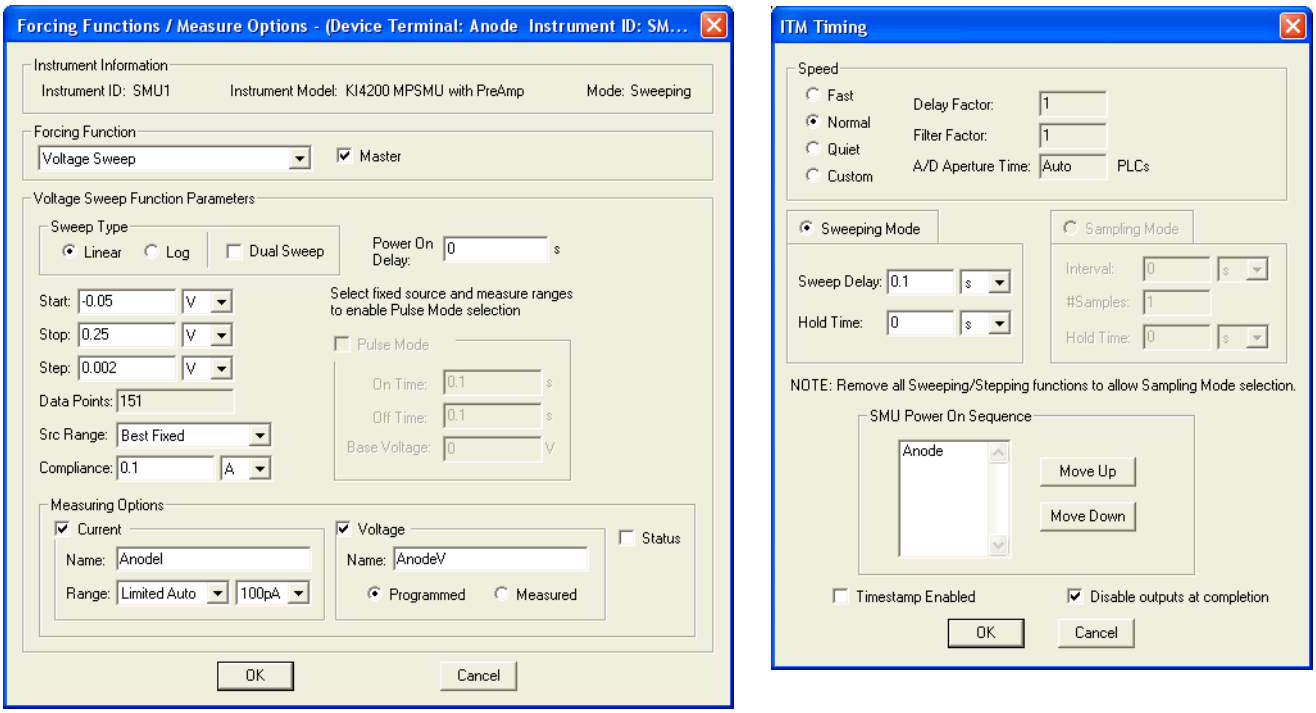
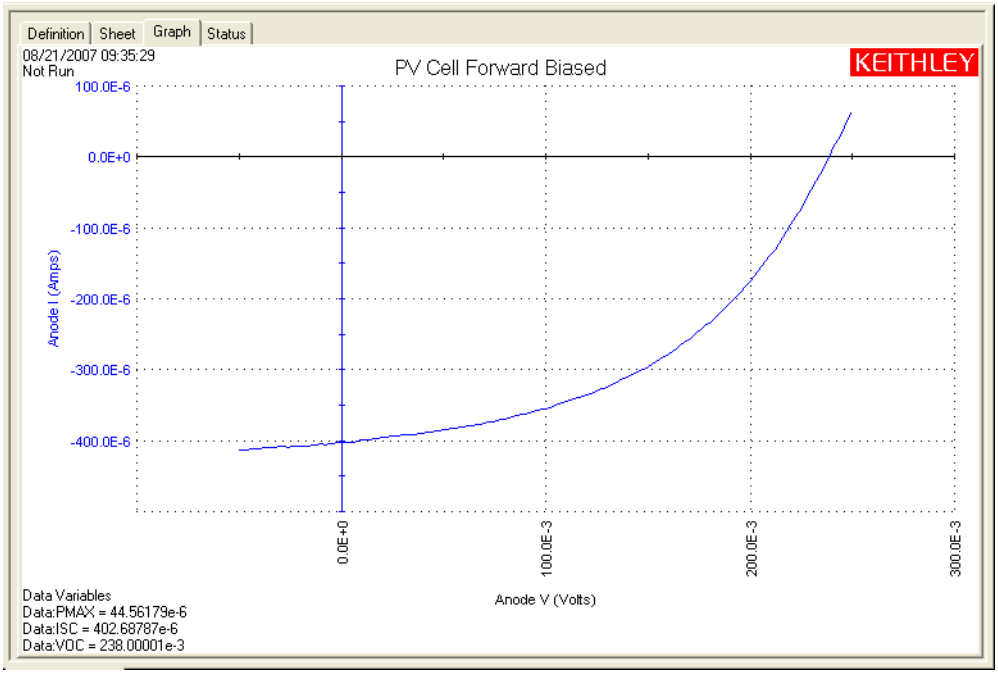


Figure 15-157
Graph tab (fwd-ivsweep ITM)



rev-ivsweep ITM

Test summary

NOTE *Make sure the PV cell is connected to the Model 4200-SMU (see Connections).*

This ITM uses a SMU to perform a reversed-biased voltage sweep on a PV cell. Current is measured on each step of the sweep. An I (absolute value) versus V graph is generated from the collected data.

When the test is started, the SMU sweeps from 0 V to -600 mV in -5 mV steps. A total of 121 current measurements are performed.

Force, measure, and timing settings

In the Project Navigator, double-click the **rev-ivsweep** ITM to display the Definition tab. The force-measure and timing settings for the Model 4200-SMU are shown in Figure 15-158 and Figure 15-159. The GNDU is selected from the Definition tab.

NOTE *Details on KITE configuration for SMUs are provided in Section 6.*

Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

- AnodeI Measured current.
- AnodeV Forced voltage.

Graph

The graph is displayed in the Graph tab (see Figure 15-160)

Figure 15-158
Definition tab (rev-ivsweep ITM)

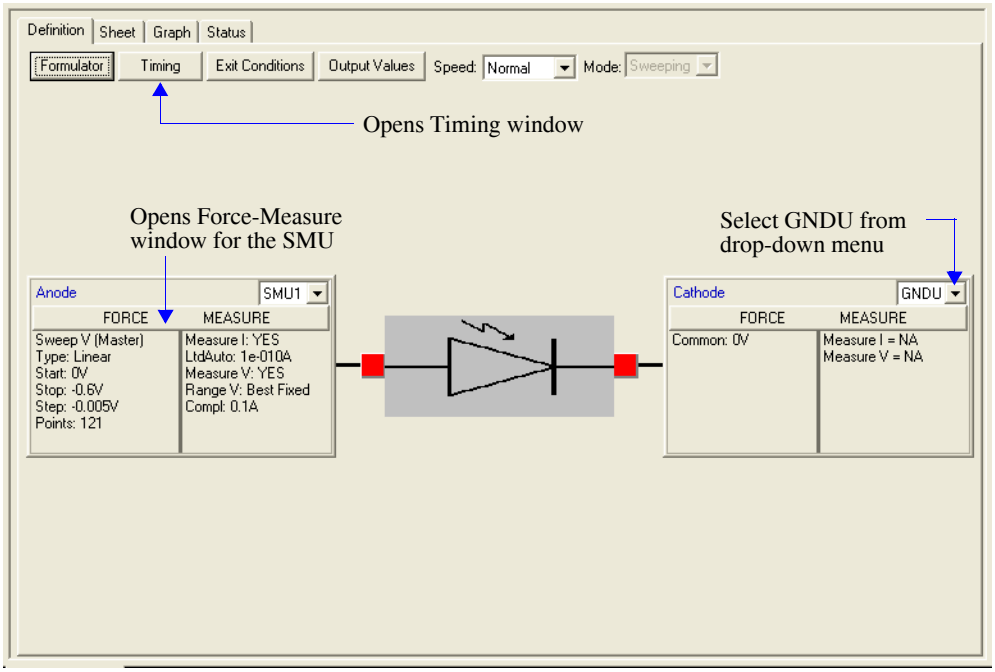
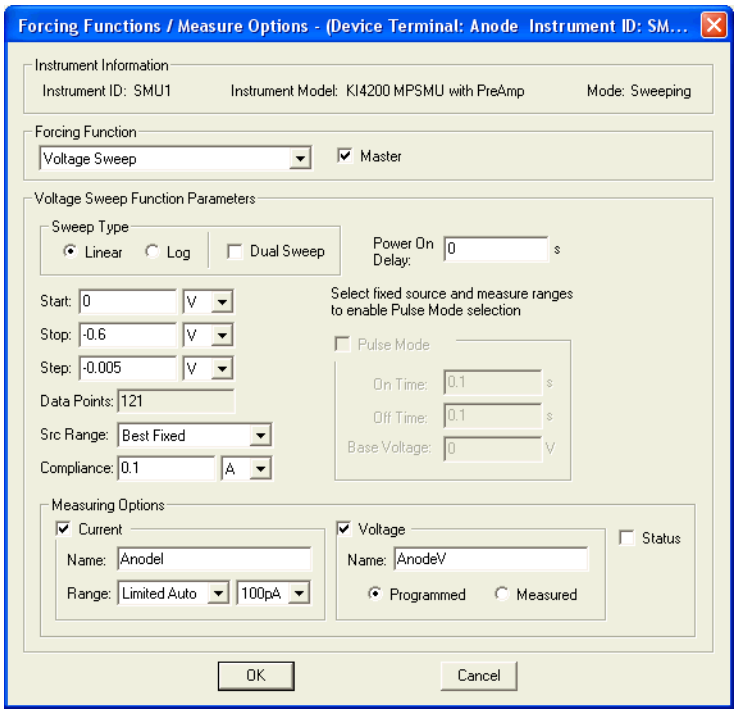


Figure 15-159
Forcing Functions / Measure Options and Timing windows (rev-ivsweep ITM)



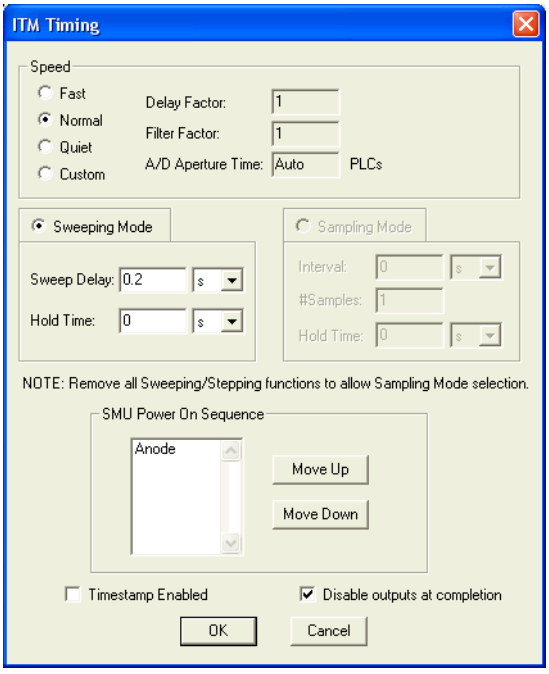
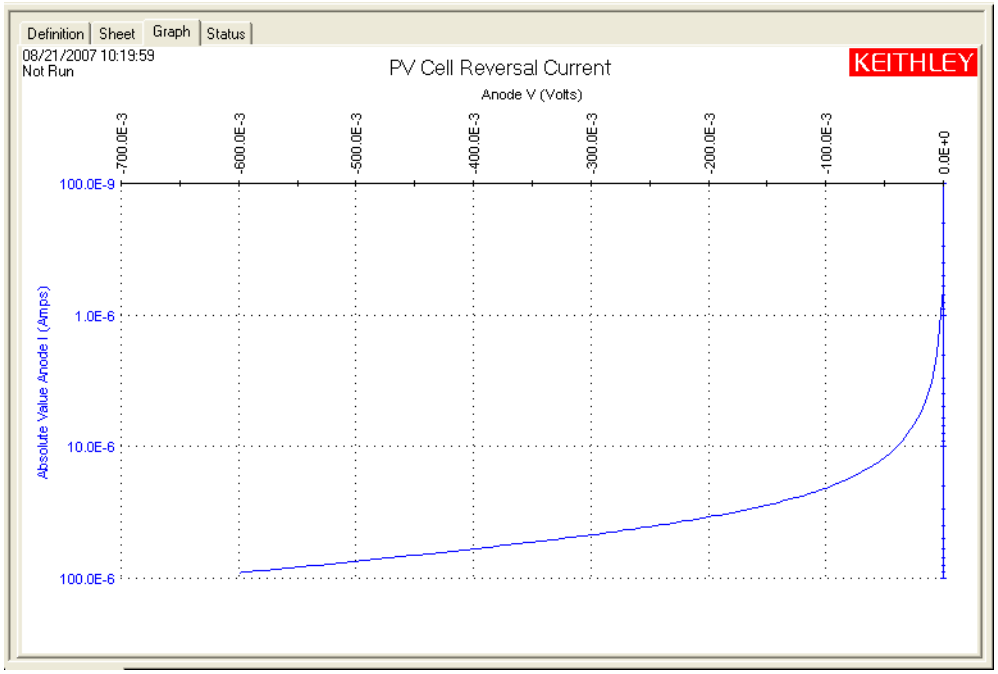


Figure 15-160
Graph tab (rev-ivsweep ITM)



cvsweep ITM

Test summary

NOTE *Make sure the PV cell is connected to the Model 4200-CVU (see Connections).*

This ITM uses a CVU to perform a forward-biased voltage sweep on a PV cell. Capacitance is measured on each step of the sweep. A C versus V graph is generated from the collected data.

Parameter settings

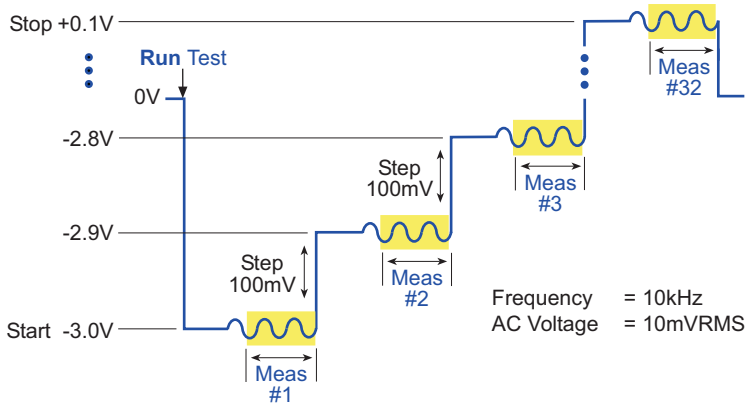
The default parameter settings for this test are listed in Table 15-40. The voltage sweep used for this test is shown in Figure 15-161.

Table 15-40
Parameter settings (cvsweep ITM)

Force (Figure 15-163)	**Measure** (Figure 15-163)	**Timing** (Figure 15-162)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -3 V Stop: 0.1 V Step: 0.1 V AC Drive Conditions Frequency: 10 kHz AC Voltage: 10 mV Data Points: 32	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.1 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-163)	**CVU Compensation** [3] (Figure 15-163)	
Anode: DC Source V Anode: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-161
Linear voltage sweep (cvsweep ITM)



In the Project Navigator, double-click the **cvsweep** ITM to display the Definition tab, which is shown in Figure 15-162.

Figure 15-162
Definition tab (cvsweep)

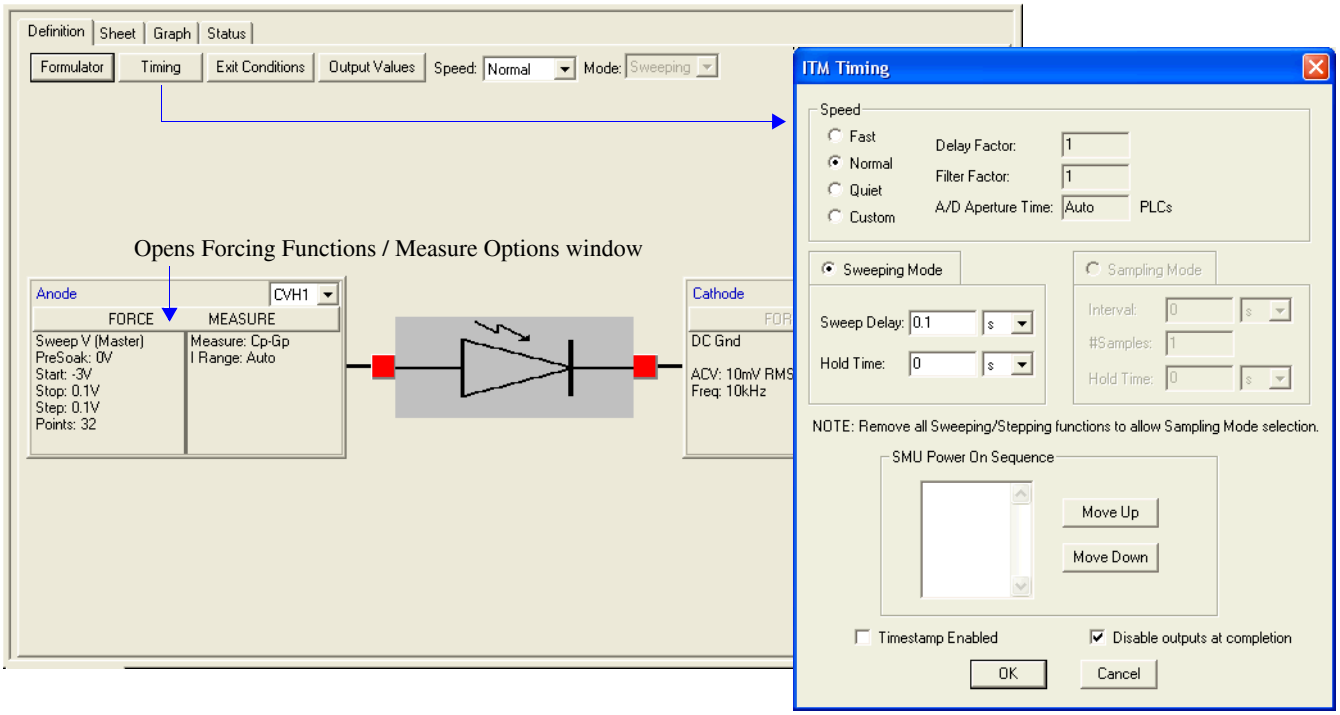
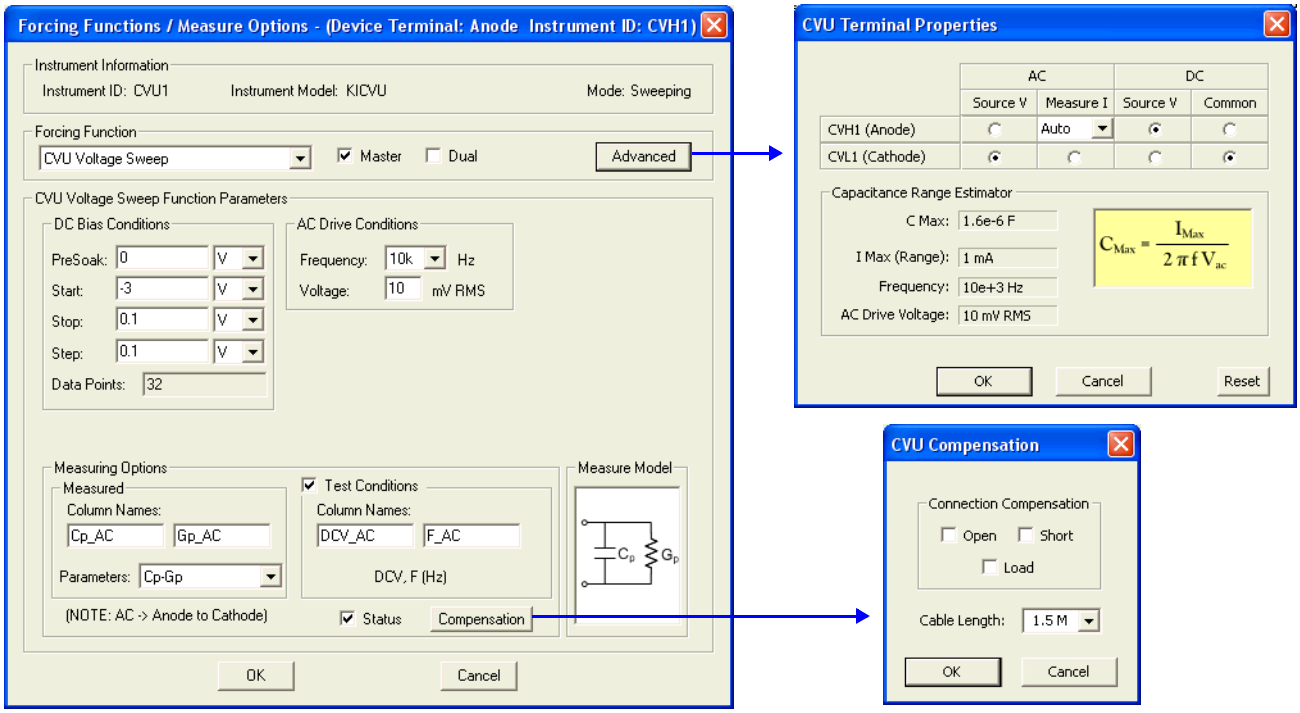


Figure 15-163
Forcing Functions / Measure Options window (cvsweep ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

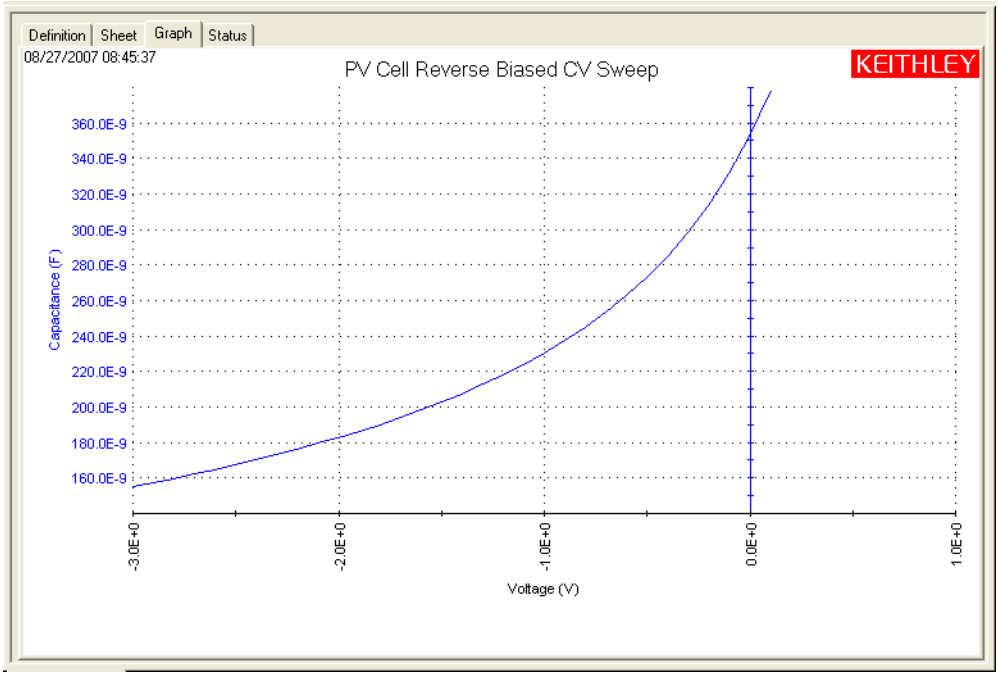
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.

Note: AC = anode-to-cathode.

Graph

The graph is displayed in the Graph tab (see Figure 15-164).

Figure 15-164
Graph tab (cvsweep ITM)



C-2vsV ITM

ITM

Test summary

NOTE *Make sure the PV cell is connected to the Model 4200-CVU (see Connections).*

This ITM uses a CVU to perform a forward-biased voltage sweep on a PV cell. Capacitance is measured on each step of the sweep. A graph ($1/C^2$ and C versus V) is generated from the collected data. The formulator calculates $1/C^2$ and the doping density.

Parameter settings

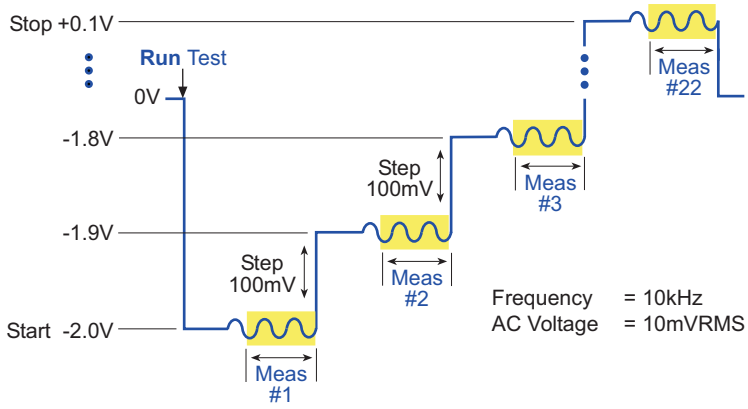
The default parameter settings for this test are listed in Table 15-41. The voltage sweep used for this test is shown in Figure 15-165.

Table 15-41
Parameter settings (C-2vsV ITM)

Force (Figure 15-167)	**Measure** (Figure 15-167)	**Timing** (Figure 15-166)
CVU Voltage Sweep: DC Bias Conditions PreSoak: 0 V Start: -2 V Stop: 0.1 V Step: 0.1 V AC Drive Conditions Frequency: 10 kHz AC Voltage: 10 mV Data Points: 22	Cp-Gp DCV Frequency I Range: Auto [1] Status: Enabled	Speed: Normal Mode: Sweeping Sweep Delay: 0.1 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-167)	**CVU Compensation** [3] (Figure 15-167)	
Anode: DC Source V Cathode: AC Measure I (Auto)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-165
Linear voltage sweep (C-2vsV ITM)



In the Project Navigator, double-click the **C-2vsV** ITM to display the Definition tab, which is shown in Figure 15-166.

Figure 15-166
Definition tab (C-2vsV ITM)

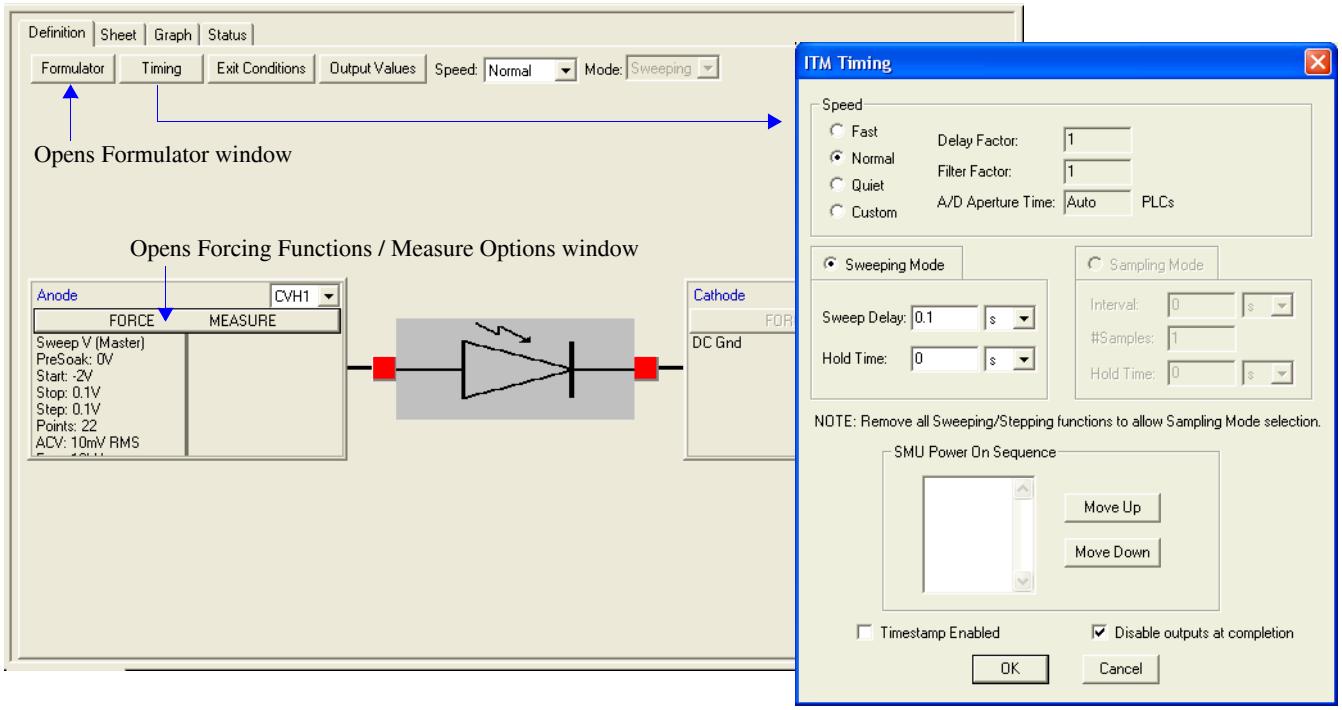
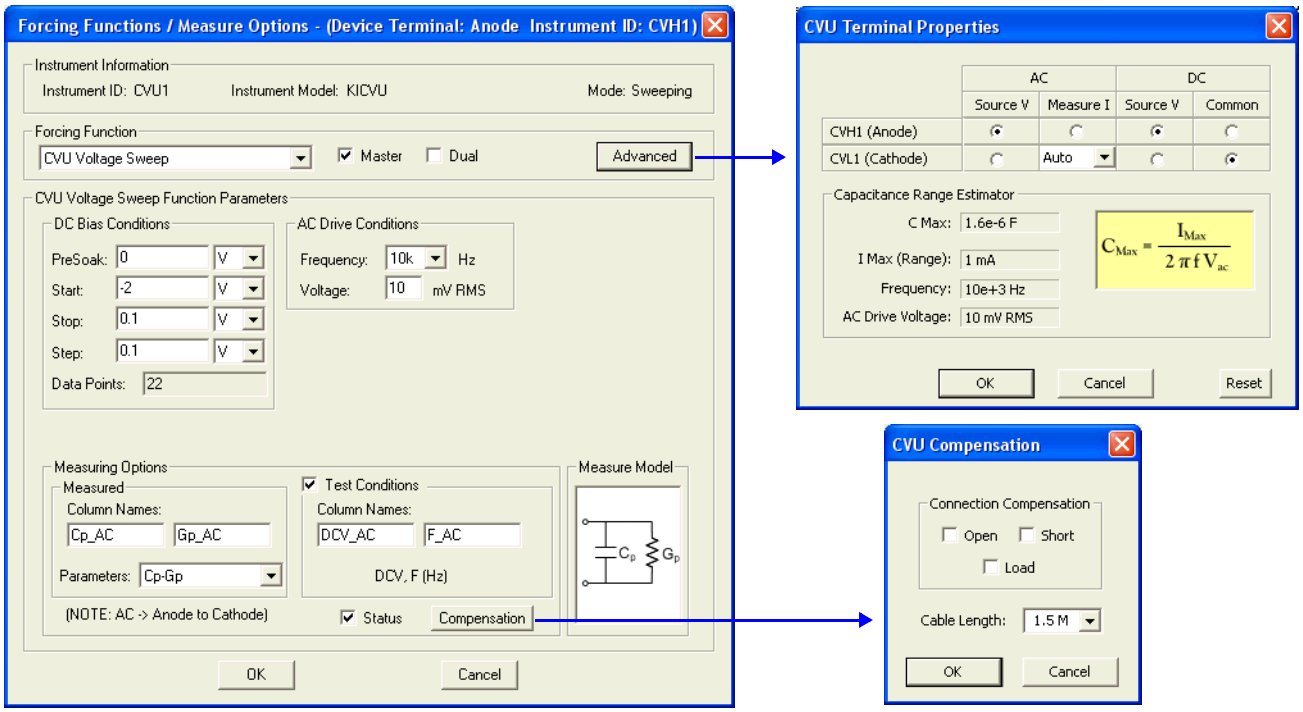


Figure 15-167
Forcing Functions / Measure Options window (C-2vsV ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

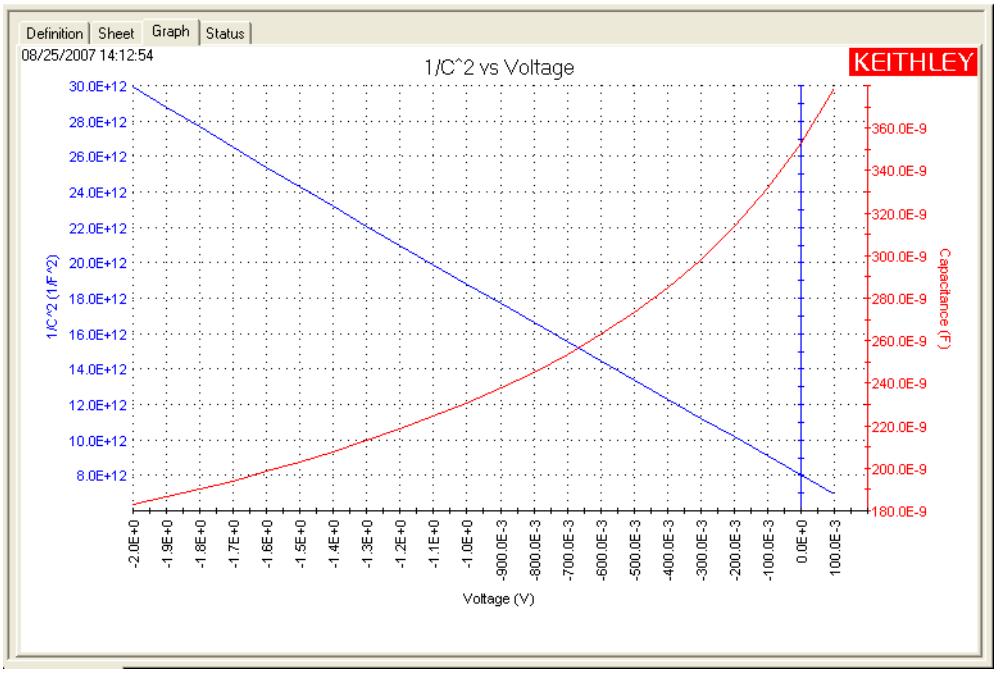
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.
- Formulas Formulator calculation results.

Note: AC = anode-to-cathode.

Graph

The I/C^2 and C vs. V graphs are displayed in the Graph tab (see Figure 15-168).

Figure 15-168
Graph tab (C-2vsV ITM)



cfsweep ITM

Test summary

NOTE *Make sure the PV cell is connected to the Model 4200-CVU (see Connections).*

This ITM performs a frequency sweep, measuring capacitance at each frequency point, and then generates a C versus F graph.

Parameter settings

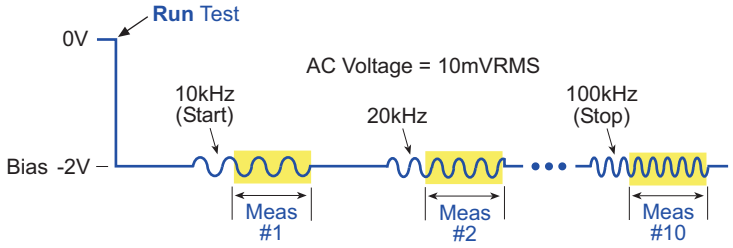
The default parameter settings for this test are listed in Table 15-42, and the voltage sweep used for this test is shown in Figure 15-169.

Table 15-42
Parameter settings (cfsweep ITM)

Force (Figure 15-171)	**Measure** (Figure 15-171)	**Timing** (Figure 15-170)
CVU Frequency Sweep: DC Bias Conditions PreSoak: 0 V DC Bias: -2 V AC Drive Conditions Start Frequency: 10 kHz Stop Frequency: 100 kHz AC Voltage: 10 mV Data Points: 10	Cp-Gp DCV Frequency I Range: 1 mA [1] Status: Enabled	Speed: Quiet Mode: Sweeping Sweep Delay: 0.2 s Hold Time: 0 s Timestamp: Disabled Outputs at completion: Disable
Advanced (Terminal Properties) [2] (Figure 15-171)	**CVU Compensation** [3] (Figure 15-171)	
Anode: DC Source V Cathode: AC Measure I (1 mA)	Enable compensation as required for the test. Set the cable length being used.	

1. Current measure range is an advanced setting (set in the Terminal Properties window).
2. For details, see the User's Manual, Advanced settings (terminal properties), page 3-16.
3. For details, see Connection compensation.

Figure 15-169
Frequency sweep (cfsweep ITM)



In the Project Navigator, double-click the **cfsweep** ITM to display the Definition tab, which is shown in Figure 15-170.

Figure 15-170
Definition tab (cfsweep ITM)

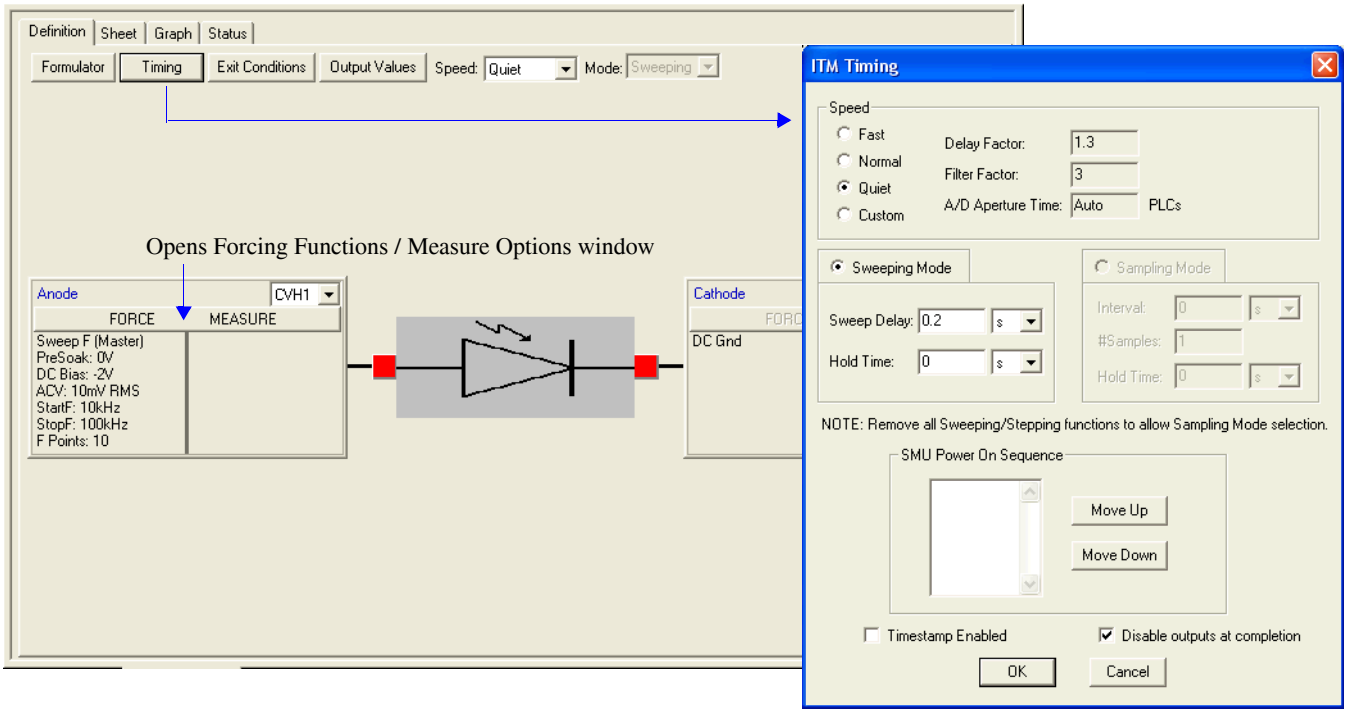
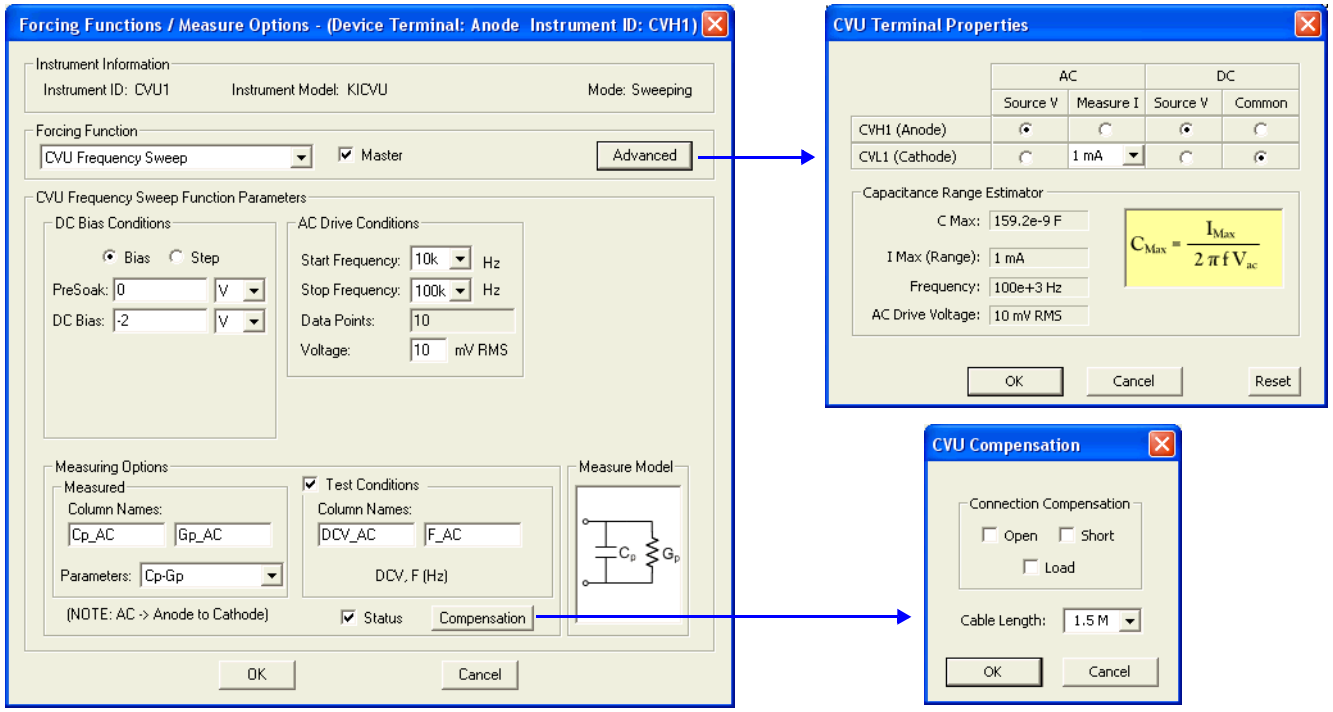


Figure 15-171
Forcing Functions / Measure Options window (cfsweep ITM)



Data sheet

Test data is displayed in the Sheet tab (example shown in Figure 15-175):

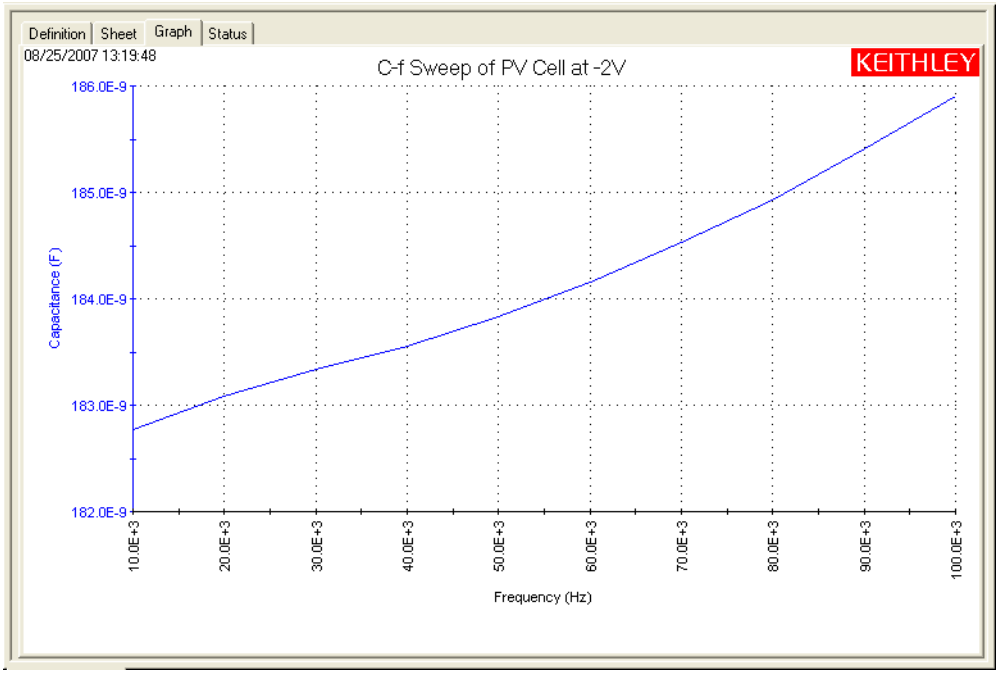
- Cp_AC Measured parallel capacitance.
- Gp_AC Measured conductance.
- DCV_AC Forced DC bias voltage.
- F_AC Forced test frequency.
- CVUS1 Status code for each measurement. Rows highlighted in blue indicate a fault. For details, see CVU measurement status.

Note: AC = anode-to-cathode.

Graph

The graph is displayed in the Graph tab (see Figure 15-172).

Figure 15-172
Graph tab (cfsweep ITM)



Default

Project Plan

Project summary

The default project has been a basic staple for the Model 4200-SCS system since its inception. The tests in this project are intended to represent the most common device tests that a typical user might regularly perform.

The original default project consisted of tests (ITMs) using Model 4200-SMUs, and is documented in Section 6 (KITE). The project has been modified to include three ITMs for C-V testing: cv-nmosfet, cv-diode, cv-cap.

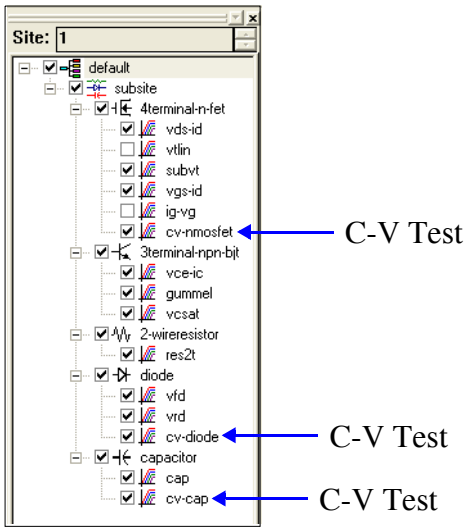
Opening the project plan

This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back (up one level) to the **Projects** folder.
3. Double-click the **default** folder.
4. Open the **default** project.

The default project plan is shown in Figure 15-173.

Figure 15-173
default project plan



Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

cv-nmosfet ITM

This ITM performs a capacitance measurement at each step of a linear voltage sweep, and generates a Capacitance versus Voltage graph.

This ITM is identical to the G_SDM ITM used in the CVU_MOSFET project. The only difference for the test are the connections for the MOSFET. For the G_SDM test, the drain and source terminals are connected to the bulk terminal. For the cv-nmosfet test, the drain and source terminals are left

unconnected on the Definition tab. When the actual device is measured, the capacitance as a function of voltage is measured between the gate and the source/drain/bulk.

NOTE *For details on configuring and running this ITM, refer to the G_SDB ITM in the following project: CVU_MOSFET.*

Also see Key concepts for MOSFET testing for additional information on testing this device.

cv-diode ITM

The junction capacitance is measured as a function of the DC bias voltage while the junction is forward biased. This test performs a capacitance measurement at each step of a user-configured linear voltage sweep. From the acquired C-V data, parameters are calculated using the formulator.

This ITM is identical to the CVsweep ITM used in the CVU_PNjunction project.

NOTE *For details on configuring and running this ITM, refer to the CVsweep ITM in the following project: CVU_PNjunction.*

Also see Key concepts for PN junction and Schottky diode testing for additional information on testing this device.

cv-cap ITM

This ITM performs a voltage sweep measuring capacitance on each step, and generates a C versus V graph. It also calculates noise.

This ITM is identical to the cv-10 pF ITM used in the CVU_Capacitor project.

NOTE *For details on configuring and running this ITM, refer to the cv-10 pF ITM in the following project: CVU_Capacitor.*

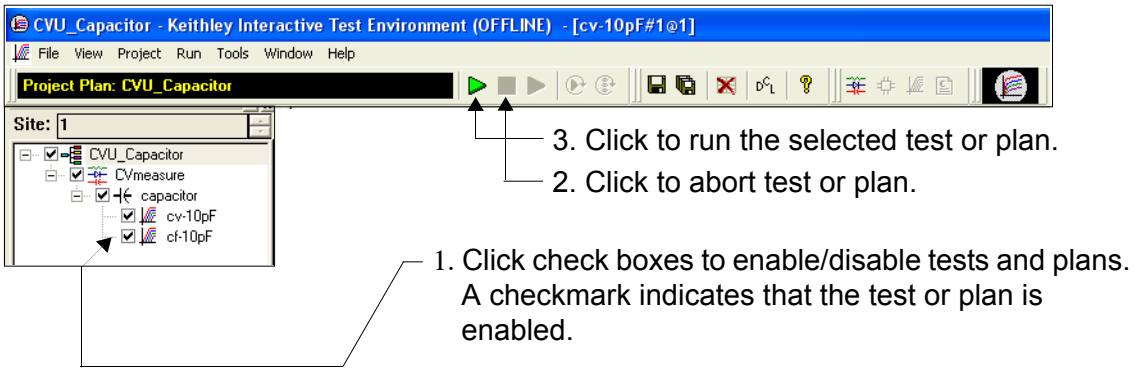
Running project plan tests

A single ITM or UTM can be run, or multiple tests can be executed by running a device plan or project plan. Only an enabled test or plan can be run.

Perform the following steps to run an individual test or all the tests in the project:

1. Connect the Model 4200-CVU to the device under test (DUT) as explained in the Connections documentation for each project plan.
2. Perform Connection compensation to correct for offset and gain errors caused by the connections.
3. Configure the tests as explained in the documentation for the ITMs and UTMs.
4. Perform the steps in Figure 15-174 to run a single test or all the tests in the project plan. The project plan navigator shown in Figure 15-174 shows all tests and plans enabled (checked). To run the second ITM only, select the **cf-10 pf** ITM, and then run it. To run the entire project plan (all tests), select the **CVU_Capacitor** project plan, and then run it.

Figure 15-174
Running a test or plan



5. Analyze the test data in the Sheet and Graph tabs. Most, but not all ITMs generate a graph. Figure 15-175 shows an example of a Sheet tab. The graph generated by an ITM is provided with the documentation for the ITM.

Figure 15-175
Sheet tab for test data

Definition | Sheet | Graph | Status

Formulas: Save As...

	A	B	C	D	E	F	G
1	Time	Cp_BE	Gp_BE	DCV_BE	F_BE	CVU1S	AVG_CAP
2	1.1137E+0	7.0398E-12	388.5131E-9	000.0000E-3	1.0000E+6	00000000	7.0405E
3	1.4545E+0	7.0404E-12	389.0903E-9	000.0000E-3	1.0000E+6	00000000	
4	1.7953E+0	7.0420E-12	392.8598E-9	000.0000E-3	1.0000E+6	00000000	
5	2.1361E+0	7.0400E-12	390.9803E-9	000.0000E-3	1.0000E+6	00000000	
6	2.4769E+0	7.0403E-12	393.9250E-9	000.0000E-3	1.0000E+6	00000000	
7	2.8177E+0	7.0396E-12	386.6811E-9	000.0000E-3	1.0000E+6	00000000	
8	3.1585E+0	7.0404E-12	394.2841E-9	000.0000E-3	1.0000E+6	00000000	
9	3.4993E+0	7.0395E-12	389.7543E-9	000.0000E-3	1.0000E+6	00000000	
10	3.8401E+0	7.0403E-12	381.7254E-9	000.0000E-3	1.0000E+6	00000000	
11	4.1809E+0	7.0404E-12	388.7875E-9	000.0000E-3	1.0000E+6	00000000	
12	4.5217E+0	7.0411E-12	383.2037E-9	000.0000E-3	1.0000E+6	00000000	
13	4.8625E+0	7.0398E-12	388.9938E-9	000.0000E-3	1.0000E+6	00000000	
14	5.2033E+0	7.0404E-12	393.7589E-9	000.0000E-3	1.0000E+6	00000000	
15	5.5441E+0	7.0406E-12	384.0209E-9	000.0000E-3	1.0000E+6	00000000	
16	5.8849E+0	7.0421E-12	395.0309E-9	000.0000E-3	1.0000E+6	00000000	
17	6.2257E+0	7.0417E-12	398.3097E-9	000.0000E-3	1.0000E+6	00000000	
18	6.5665E+0	7.0402E-12	385.5740E-9	000.0000E-3	1.0000E+6	00000000	
19	6.9073E+0	7.0415E-12	380.7765E-9	000.0000E-3	1.0000E+6	00000000	
20	7.2481E+0	7.0409E-12	381.5616E-9	000.0000E-3	1.0000E+6	00000000	
21	7.5888E+0	7.0398E-12	387.1396E-9	000.0000E-3	1.0000E+6	00000000	

Data | Calc | Settings

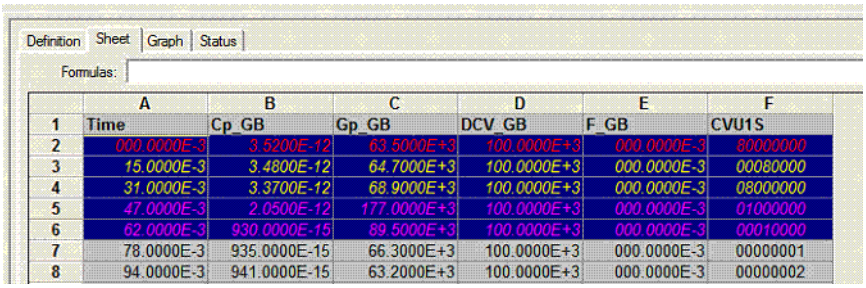
CVU measurement status

Many tests (ITMs) provide status information for the Model 4200-CVU measurements in the Sheet tab of KITE. The data column for the 32-bit status codes is labeled CVU1S. CVU status code indicates the I measure range for each impedance measurement and flags any faults (errors).

When a measurement fault occurs, the entire row of data related to the measurement will be highlighted in blue (see Figure 15-176). The data values in the flagged data row will be color-coded to identify the fault type as follows:

Red = Measurement timeout
Magenta = Measurement overflow
Yellow = ABB not locked

Figure 15-176
Status tab showing faults (example)



	A	B	C	D	E	F
1	Time	Cp_GB	Gp_GB	DCV_GB	F_GB	CVU1S
2	000.0000E-3	3.5200E-12	63.5000E+3	100.0000E+3	000.0000E-3	80000000
3	15.0000E-3	3.4800E-12	64.7000E+3	100.0000E+3	000.0000E-3	00080000
4	31.0000E-3	3.3700E-12	68.9000E+3	100.0000E+3	000.0000E-3	08000000
5	47.0000E-3	2.0500E-12	177.0000E+3	100.0000E+3	000.0000E-3	01000000
6	62.0000E-3	930.0000E-15	89.5000E+3	100.0000E+3	000.0000E-3	00010000
7	78.0000E-3	935.0000E-15	66.3000E+3	100.0000E+3	000.0000E-3	00000001
8	94.0000E-3	941.0000E-15	63.2000E+3	100.0000E+3	000.0000E-3	00000002

Placing the cursor on a flagged CVU1S cell will open a window that summarizes the fault.

When a measurement fault occurs, a message appears in the upper left-hand corner of the graph.

Status codes

The 16 basic codes used for Model 4200-CVU measurement status are listed in Table 15-43. Each code is represented as a 32-bit hexadecimal value (0x). As shown in the table, the mr value is last two digits of each code.

mr = 00 Lowest range (1µA) used for the impedance measurement
= 01 Middle range (30µA) used for the impedance measurement
= 02 Highest range (1 mA) used for the impedance measurement

Table 15-43
CVU measurement status codes (CVU1S)

#	Code	Description
1	000000mr	No faults
2	8xxx00mr	Measurement timeout occurred
3	01xx00mr	CVH1 current measurement overflow
4	02xx00mr	CVH1 voltage measurement overflow
5	03xx00mr	CVH1 I and V measurement overflow
6	08xx00mr	CVH1 ABB not locked
7	09xx00mr	CVH1 ABB not locked, I measurement overflow
8	0Axx00mr	CVH1 ABB not locked, V measurement overflow
9	0Bxx00mr	CVH1 ABB not locked, I and V measurement overflow
10	xx0100mr	CVL1 I measurement overflow
11	xx0200mr	CVL1 V measurement overflow
12	xx0300mr	CVL1 I and V measurement overflow
13	xx0800mr	CVL1 ABB not locked
14	xx0900mr	CVL1 ABB not locked, I measurement overflow
15	xx0A00mr	CVL1 ABB not locked, V measurement overflow
16	xx0B00mr	CVL1 ABB not locked, I and V measurement overflow

Example

CVU status code: 08000001

- The 08 indicates that the CVHI ABB was not locked when the measurement was performed.
- The 01 indicates that the middle range (30µA) was used for the impedance measurement.

Measurement status notes

NOTE *Whenever a fault occurs, run the Confidence Check utility before performing any other troubleshooting actions (see Confidence Check for details).*

Measurement timeout: Indicates that the measurement was not received after a set time (total aperture).

- This timeout error may indicate that there is an issue with the Model 4200-CVU card. Try resetting the hardware and running the project test again. If this error reoccurs, contact Keithley Instruments.
- The hardware can be reset from Windows® in one of two ways:

  a. Use the Run window: Click **Start** and then click **Run**. Type in **resethw** and click **OK**.

  OR

  b. Use the Command Prompt: Click **Start** > **Programs** > **Accessories** > **Command Prompt**. Type **resethw** and press the **Enter** key on the keyboard.

I measurement overflow:

- Try a higher I measure range (or Auto).
- Try a lower AC drive voltage.

V measurement overflow: Try a lower DC bias voltage.

ABB not locked: Auto Balance Bridge not locked when measurement was performed. The readings (and calculation results) may not be accurate. Causes for this fault are explained in the Measurement overview.

LPT library function reference

The LPT functions that pertain to the Model 4200-CVU are listed in Table 15-44. LPT function details follow the table and are presented in alphabetical order.

Table 15-44
LPTLib function listing for C-V testing

Group		Function	
CVU	General	devclr	Device clear: Turn off DCV bias
		devint	Device initialize
		forcev	Set DC bias voltage
		getstatus	Query status of the instrument
		measf	Return sourced frequency for a single measurement
		meast	Return timestamp for a single measurement
		measv	Return the sourced DC bias voltage for a single measurement
		measz	Measure and return impedance for a single measurement
		rangei	Set measure range
		setauto	Set CVU to auto measure range
		setfreq	Set the AC drive frequency
		setlevel	Set the AC drive voltage
		setmode	Set operating parameters
	Sweeping	adelay	Specify an array of delay values for the asweepv function
		asweepv	Perform DC voltage array sweep
		dsweepf	Perform dual frequency sweep
		dsweepv	Perform dual DC voltage sweep
		rtfary	Acquire array of force values for a sweep
		smeasf	Return the sourced frequencies for a sweep
		smeasfRT	Return the sourced frequencies (in real time) for a sweep
		smeast	Return timestamps for measurements in a sweep
		smeastRT	Return timestamps (in real time) for measurements in a sweep
		smeasv	Return the sourced DC bias voltages for a sweep
		smeasvRT	Return the sourced DC bias voltages (in real time) for a sweep
		smeasz	Measure and return impedances for a sweep
		smeaszRT	Measure and return (in real time) impedances for a sweep
		sweepf	Perform frequency sweep
		sweepv	Perform DC voltage sweep

adelay

Purpose Sets an array of delay values for asweepv.

NOTE *This function can be used with any of the asweepX functions (see asweepX functions in Section 8 for details). The following information pertains specifically to the Model 4200-CVU.*

Format

```
int adelay(long numberOfPoints, double *delayArray);
```

numberOfPoints Total number of sweep points

delayArray An array of delay values (in seconds)

Remarks This function is used to define an array of delay values for the points in a voltage array sweep (asweepv). Each delay in the array is added to the delay time specified in asweepv. For example, if the array contained four delays (0.04 s, 0.05 s, 0.06 s, and 0.07 s) and the delay time specified in asweepv is 0.1 s, then the resulting delays are (0.14 s, 0.15 s, 0.16 s, and 0.17 s).

The number of delay values must match the number of points in the voltage array sweep. Example: Assume `asweepv` is configured to sweep four points, and the, and the following delay times need to be set: 0.5 s, 0.25 s, 0.5 s, 0.25 s (in that order). With the delay time for `asweepv` set for 0 s, the array for the `adelay` function would be configured as follows:

```
delayArray(0) = 0.5
delayArray(1) = 0.25
delayArray(2) = 0.5
delayArray(3) = 0.25
```

Example Programming example #5 shows how to set up an array of delay times for a voltage array sweep.

asweepv

Purpose Performs a DC voltage sweep using an array of voltage values.

> ***NOTE*** *The following supplemental information on the voltage array sweep pertains specifically to the Model 4200-CVU. See asweepX functions in Section 8 for details.*

Format

```
int asweepv(INSTR_ID instid, long numberOfPoints, double
delayTime, double *forceArray);
```

	Instrument ID of the Model 4200-CVU: CVU1
	Total number of sweep points (1 to 4096)
delayTime	Delay time before each measurement (0 to 999 s)
forceArray	Array of DC voltage values

Remarks This function is used to perform a DC voltage sweep using an array of voltage values. The number of voltage values in the array must match the *numberOfPoints* parameter value.

Figure 15-177 shows an example of a voltage array sweep using the following voltage values: 1 V, -2 V, 3 V, -4 V. The voltage array would be configured as follows:

```
forceArray(0) = 1
forceArray(1) = -2
forceArray(2) = 3
forceArray(3) = -4
```

The *delayTime* parameter sets the user-programmed delay before each measurement. Note that there is an additional inherent system delay (SD) that occurs at the start of each step.

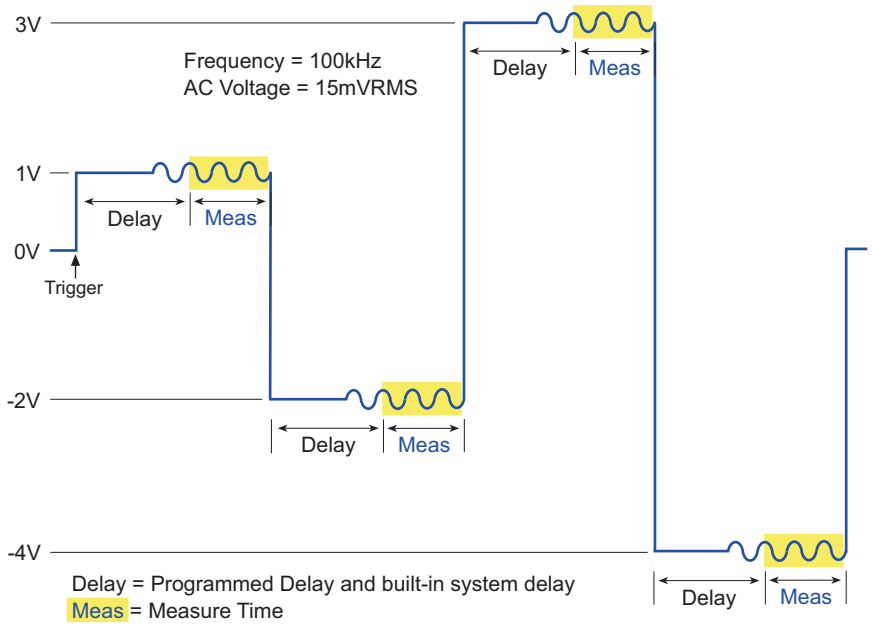
If different delay times are needed in the sweep, an array of delay time values can be set to adjust the delay times at each step (see `adelay` for details).

Use the `setfreq` and `setlevel` functions to set the AC drive frequency and voltage for the sweep.

See also `dsweepf`, `dsweepv`, `sweepf`, `sweepv`

Example Programming example #4 performs the voltage array sweep shown in Figure 15-177.

Figure 15-177
Voltage array sweep example



devclr

Purpose Sets all sources to a zero state.

Format

```
int devclr(void);
```

Remarks This function clears all sources sequentially in the reverse order from which they were originally forced. For C-V testing, this function will turn off the DC bias voltage. See `devclr` function in Section 8 for more information.

`devclr` is implicitly called by `devint`.

See also `devint`

devint

Purpose Resets the instruments and clears the system by opening all relays and disconnecting the pathways. Meters and sources are reset to the default states.

Format

```
int devint(void);
```

Remarks This function will reset all instruments in the system to their default states. The Model 4200-CVU returns to the following states:

- 30 mV RMS AC signal
- 0 V DCV bias
- 100 kHz frequency
- Auto range
- Cable length correction set to 0 M.
- Open/Short/Load compensation disabled.

See `devint` function in Section 8 for more information.

See also `devclr`

dsweepf

Purpose Performs a dual frequency sweep.

NOTE *Use the sweepf function to perform a single frequency sweep.*

Format

```
int dsweepf(INSTR_ID instid, double startf, double stopf,
long *NumPts, double delaytime);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
startf	Initial frequency for the sweep
stopf	Final frequency for the first sweep
NumPts	Variable to receive the number of points sourced during the sweep
delayTime	Delay before each measurement (0 to 999 s)

Remarks This function is used to perform a dual frequency sweep (see Figure 15-178). The Model 4200-CVU provides test frequencies from 10 kHz to 10 MHz in the following steps:

- 10 kHz to 100 kHz in 10 kHz steps
- 100 kHz to 1 MHz in 100 kHz steps
- 1 MHz to 10 MHz in 1 MHz steps

The frequency points to sweep are set using the *startf* and *stopf* parameters. If an entered value is not a supported frequency, the closest supported frequency will be selected (for example, 15kHz input will select 20 kHz). If a specified frequency is equidistant from two adjacent frequencies, LPTLIB will round up to the higher frequency. The sweep can step forward (low frequency to high frequency) or it can step in reverse (high frequency to low frequency).

When the sweep is started, the CVU will step through all the supported frequency points from start to stop for the first sweep, and then repeat (in the reverse direction) from stop to start for the second sweep. For example, if the start frequency is 800 kHz and the stop frequency is 3 MHz, the CVU will step through the following frequency points:

800 kHz, 900 kHz, 1 MHz, 2 MHz, 3 MHz, 3 MHz, 2 MHz, 1 MHz, 900 kHz, 800 kHz

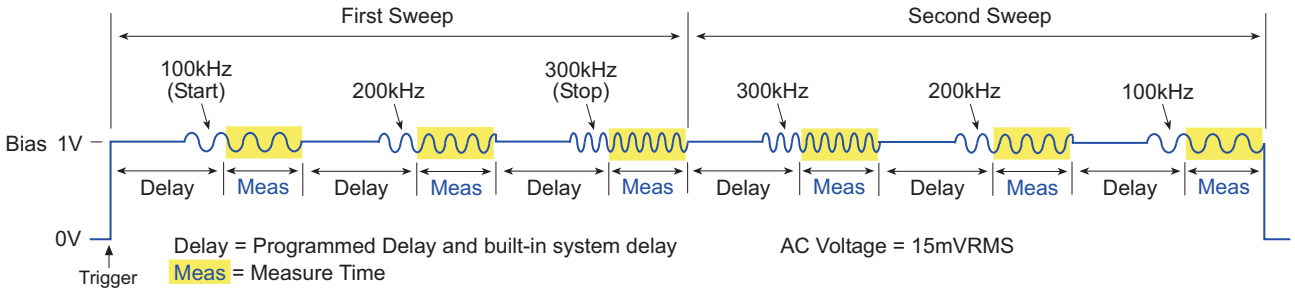
The total number of sweep points will be returned in the *NumPts* parameter. For the above example, *NumPts* will be assigned a value of 10.

The *delayTime* parameter sets the delay that occurs before each measurement. Note that there is an inherent system overhead delay (SD) on each frequency step of the sweep.

Use the forcev function to set the DC bias level and setlevel function to set the AC drive voltage.

See also asweepv, dsweepv, sweepf, sweepv

Figure 15-178
Dual frequency sweep example



dsweepv

Purpose Performs a dual linear staircase voltage sweep.

NOTE *Use the sweepv function to perform a single linear staircase voltage sweep.*

Format

```
int dsweepv(INSTR_ID instid, double startv, double stopv,
long numSteps, double delaytime);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
startv	Initial force value for the sweep (-30 V to 30 V)
stopv	Final force value for the first sweep (-30 V to 30 V)
numSteps	Sets the number of points in the sweep (1 to 4096)*
delaytime	Delay before each measurement (0 to 999 s)

* This number describes the first "half" of the sweep. For example, to do a dual sweep from 1 V to 10 V and back down in 1 V steps, *numSteps* would be set to 10. The end result will be a 20-point sweep (10 up and 10 down).

Remarks This function is used to perform a dual staircase sweep (see Figure 15-179). The linear step size to sweep is set using the *startv*, *stopv* and *NumPoints* parameters. The linear step size for the sweep is then calculated as follows:

StepSize (in volts) = (*stopv* – *startv*) / (*numSteps*)

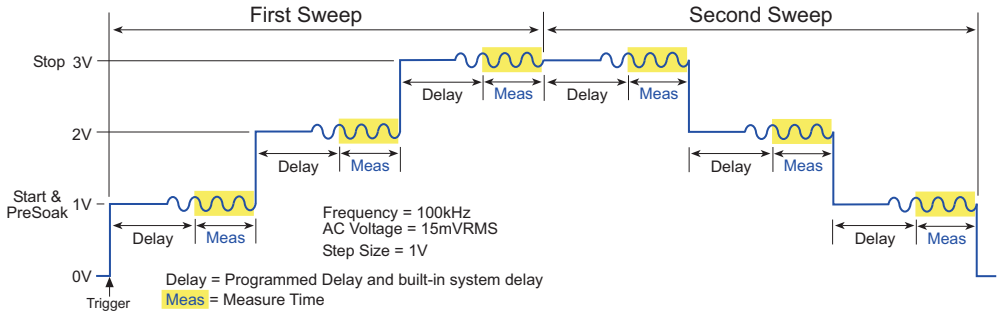
The first sweep can step forward (low voltage to high voltage) or it can step in reverse (high voltage to low voltage). After performing the first sweep, the second sweep will repeat in the reverse direction. For example, if configured to sweep from 1 V to 10 V, the second sweep will start at 10 V and step down to 1 V.

The *delayTime* parameter sets the Delay that occurs before each measurement. Note that there is an inherent system overhead delay (SD) on each step of the sweep.

Use the and functions to set the AC drive frequency and voltage for the sweep.

See also , , ,

Figure 15-179
Dual voltage sweep example



forcev

Purpose Sets the DC bias voltage level.

Format

```
int forcev(INSTR_ID instid, double voltage);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
voltage	DC bias voltage level (-30 V to 30 V)

Remarks This function is used to set a DC bias level for a single impedance measurement, and a frequency sweep. Use the and functions to set the AC drive frequency and AC voltage for the sweep.

The DC source operates independently of the AC source. Changes to the level and state of the DC source take effect immediately, whereas the AC frequency and source value are only utilized during `measz` operations.

Example Programming example #1 performs a single impedance measurement. Note that the `rdelay` function provides a settling time before the measurement.

getstatus

Purpose Returns various parameters pertaining to the state of the Model 4200-CVU.

Format

```
int getstatus(INSTR_ID instid, long parameter, double
*value);
```

`instid`	Instrument ID of the Model 4200-CVU: CVU1
`parameter`	Parameter to be queried. *
`value`	Returned value for the queried parameter.

* The macros for the KI_CVU parameters are defined in lptparam.h.

Parameter:	Returns:
KI_CVU_LOAD_COMPENSATE	Load compensation (ON or OFF)
KI_CVU_OPEN_COMPENSATE	Open compensation (ON or OFF
KI_CVU_SHORT_COMPENSATE	Short compensation (ON or OFF)
KI_CVU_CABLE_CORRECT	Current length setting for which the CVU card is correcting (0, 1.5 or 3).
KI_CVU_ACI_RANGE	AC current range (0 for auto range, or 1.5µA, 50µA or 1.5 mA for fixed range)

KI_CVU_ACI_PRESENT_RANGE	AC current range (1.5µA, 50µA or 1.5 mA)[1]
KI_CVU_ACV_LEVEL	AC voltage level (10 mV to 100 mV RMS)
KI_CVU_APERTURE or KI_INTGPLC	A/D aperture time (0.006 to 10.002 PLCs) Integration (NPLC= 1/aperture time)
KI_CVU_DCV_LEVEL	DC bias voltage level (-30 V to 30 V)
KI_CVU_DELAY_FACTOR	Delay factor (0 to 100)
KI_CVU_FILTER_FACTOR	Filter factor (0 to 100)
KI_CVU_FREQUENCY	Drive frequency (10 kHz to 10 MHz)
KI_CVU_MEASURE_MODEL	Impedance measure model (0 through 5; see Table 15-45)
KI_CVU_MEASURE_SPEED	Measurement speed (fast, normal, quiet or custom)
KI_CVU_MEASURE_STATUS	Measurement status (for last reading)[2]
KI_CVU_MEASURE_TSTAMP	Measurement timestamp (for last reading)

1. Returns range used for last measurement, even if on auto range.
2. The measurement status codes are listed and explained in Table 15-43.

measf

Purpose Returns the frequency sourced during a single measurement.

Format

```
int measf(INSTR_ID instid, double *freq);
```

	Instrument ID of the Model 4200-CVU: CVU1
freq	Returned frequency

Remarks This function returns the present test frequency being used for a single impedance measurement. The function is used to perform a single measurement.

NOTE *Use the smeasf or smeasfRT function to return the frequencies used for a sweep.*

meast

Purpose Returns a timestamp referenced to a measurement or a system timer.

Format

```
int meast(long timerID, double *timestamp);
```

timerID	ID of the Model 4200-CVU or timer: CVU1 or TIMER1, TIMER2, and so on.
timeStamp	Returned timestamp

Remarks This function is used acquire the timestamp of the last single measurement, or return a timestamp referenced to a system timer.

When the *timerID* parameter is set for CVU1, calling the `meast` function after the call to perform a measurement (`measz` function) will return the timestamp for that measurement.

When the *timerID* parameter is set for a timer, the meast function can be called at any time and will return a timestamp that is referenced to a system timer. The enable function is used to start the timer (starts at zero when called).

NOTE *Use the smeast or smeastRT function to acquire timestamps for a sweep.*

Example Programming example #1 acquires a timestamp for the measurement. Programming example #2 measures the execution time of the code.

measv

Purpose Returns the DC bias voltage sourced during a single measurement.

Format

```
int measf(INSTR_ID instid, double *biasV);
```

instid Instrument ID of the Model 4200-CVU: CVU1

biasV Returned DC bias voltage

Remarks This function returns the DC bias voltage presently being used for a single measurement. The function is used to perform a single measurement.

NOTE *Use the smeasv or smeasvRT function to return the DC bias voltages used for a sweep.*

measz

Purpose Performs an impedance measurement.

Format

```
int measz(INSTR_ID instid, int model, int speed, double
*result1 double *result2);
```

instid Instrument ID of the Model 4200-CVU: CVU1

model Measurement model (see Table 15-45).

speed Measure speed:

```
KI_CVU_SPEED_FAST
KI_CVU_SPEED_NORMAL
KI_CVU_SPEED_QUIET
KI_CVU_SPEED_CUSTOM
```

result1 First result of the selected measure *model*

result2 Second result of the selected measure *model*

Remarks This function performs a single impedance measurement. The parameter values for the measurement *model* are listed in Table 15-45.

Table 15-45
Measurement model parameter values

Measurement model		Parameter value	
ZTH	Impedance (Z) and phase (θ in degrees)	KI_CVU_TYPE_ZTH	or 0
RjX	Resistance and reactance	KI_CVU_TYPE_RJX	or 1
CpGp	Parallel capacitance and conductance	KI_CVU_TYPE_CPGP	or 2

Table 15-45 (continued)
Measurement model parameter values

Measurement model		Parameter value	
CsRs	Series capacitance and resistance	KI_CVU_TYPE_CSRS	or 3
CpD	Parallel capacitance and dissipation factor	KI_CVU_TYPE_CPD	or 4
CsD	Series capacitance and dissipation factor	KI_CVU_TYPE_CSD	or 5

Measurement *speed* settings:

KI_CVU_SPEED_FAST	Performs fast measurements (higher noise)
KI_CVU_SPEED_NORMAL	Selects a balance between speed and low-noise
KI_CVU_SPEED_QUIET	Performs low-noise measurements
KI_CVU_SPEED_CUSTOM	Selects custom settings: Delay factor, filter factor and aperture are set using the function.

Before calling `measz`, use the function to set the DC bias level, the function to set the AC drive frequency and the function to set the AC drive voltage.

NOTE *Use the `smeasz` or `smeaszRT` function to measure and return the impedance readings for a sweep.*

Example Programming example #1 performs a single impedance measurement.

rangei

Purpose Selects an impedance measurement range.

Format

```
int rangei(INSTR_ID instid, double range);
```

instid Instrument ID of the Model 4200-CVU: CVU1

range Impedance measure range (0, 1 µA, 30 µA or 1 mA)

Remarks This function is used to set the Model 4200-CVU to a current measure range for impedance measurements. Setting *range* to 0 selects auto range. The CVU will automatically go to the most sensitive (optimum) range to perform the measurement. This is the same as calling the `setauto` function.

The other *range* parameter values select a fixed measure range. The CVU will remain on the fixed range until auto range is enabled or the CVU is reset (`devint` called).

Example Programming example #1 uses the 1 mA measure range for the impedance measurement.

rtfary

Purpose Returns the array of force values used during the subsequent voltage or frequency sweep.

Format

```
int rtfary (double *forceArray);
```

forceArray Array of force values for voltage or frequency

Remarks This function is used to return an array of voltage or frequency force values for a sweep. This function should be sent before calling any sweep function.

NOTE Make sure that the array that will receive the sourced values is large enough, or else a memory exception will occur.

The following examples show the proper function sequence for using `rtfary`:

Example 1) Valid function sequence for voltage sweep:	Example 2) Valid function sequence for frequency sweep:
`smeasz`	`smeasz`
`smeast`	`rtfary`
`rtfary`	`dsweepf`
`dsweepv`	

Example Programming example #2 returns the array of force values for the voltage sweep.

setauto

Purpose Selects the auto measure range.

Format

```
int setauto(INSTR_ID instid);
```

instid Instrument ID of the Model 4200-CVU: CVU1

Remarks This function is used to set the Model 4200-CVU for auto range measurements. When `setauto` is called, the CVU will go to the most sensitive (optimum) range to perform the measurement. Calling `devint` also selects auto range.

The `rangei` function can also be used to enable auto range, or select a fixed measurement range. Auto range will remain enabled until a fixed range is selected.

Example Programming examples 2 through 5 use auto range for impedance measurements.

setfreq

Purpose Sets the frequency for the AC drive.

Format

```
int setfreq(INSTR_ID       , double          );
```

Instrument ID of the Model 4200-CVU: CVU1.

Frequency of the AC drive.

Remarks This function is used to set the frequency of the AC drive.The Model 4200-CVU provides test frequencies from 10 kHz to 10 MHz in the following steps:

- 10 kHz to 100 kHz in 10 kHz steps
- 100 kHz to 1 MHz in 100 kHz steps
- 1 MHz to 10 MHz in 1 MHz steps

If an entered value is not a supported frequency, the closest supported frequency will be selected (for example, 15kHz input will select 20 kHz). The `getstatus` function can be used to retrieve the selected frequency value.

AC drive (AC voltage level and frequency) does not turn on until a measurement is performed. The AC drive turns off after the measurement is completed. Note that the DC voltage source stays on for the whole test.

See also `setlevel`

Example Programming examples 1, 2, 4 and 5 use the `setfreq` function to set the AC drive frequency.

setlevel

Purpose Sets the AC drive voltage level.

Format

```
int setlevel(INSTR_ID instid, double signalLevel);
```

	Instrument ID of the Model 4200-CVU: CVU1.
signalLevel	Voltage level of the AC drive (10 mV to 100 mV RMS).

Remarks This function is used to set the voltage of the AC drive. The drive voltage can be set from 10 mV to 100 mV RMS.

AC drive (AC voltage level and frequency) does not turn on until a measurement is performed. The AC drive turns off after the measurement is completed. Note that the DC voltage source stays on for the whole test.

See also

Example All the Programming examples use the `setlevel` function to set the AC drive voltage.

setmode

Purpose Set operating modes specific to the Model 4200-CVU.

NOTE *The following information pertains specifically to the CVU. For details on using `setmode` for other instruments, see `setmode` function in Section 8.*

Format

```
int setmode(INSTR_ID instid, long modifier, double value);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
modifier	Specific operating characteristic to change (see Table 15-46)
value	Parameter value for the *modifier*.

Remarks The `setmode` function allows control over the following Model 4200-CVU operating characteristics:

- Connection compensation control (on/off) for open, short and load. When disabled, saved compensation constants will not be applied to the measurements. Whenever the connection setup has changed, connection compensation will need to be performed in order to acquire and save new compensation constants). Connection compensation is performed from KITE. For details, see Connection compensation).

- Setting for cable length correction (0 m, 1.5 m or 3 m). This setting is made from the window used to enable correction (see Enabling compensation).
- Settings (delay factor, filter factor and aperture) for KI_CUSTOM measurement speed (which is set by , or `smeaszRT`).

Table 15-46
***Modifiers* and *values* for CVU1**

Parameters		Comment
KI_CVU_OPEN_COMPENSATE KI_CVU_SHORT_COMPENSATE KI_CVU_LOAD_COMPENSATE	0 = OFF 1 = ON (for each of the three modifiers)	Enable or disable compensation constants for open, load and short.
KI_CVU_CABLE_CORRECT	0.0, 1.5, or 3.0	Cable length setting (in meters) *
KI_CVU_MEASURE_SPEED	KI_CVU_SPEED_FAST KI_CVU_SPEED_NORMAL KI_CVU_SPEED_QUIET KI_CVU_SPEED_CUSTOM	Fast measurements (higher noise) Balance between speed and low-noise Low-noise measurements Custom settings (see next modifiers)
KI_CVU_APERTURE KI_CVU_DELAY_FACTOR KI_CVU_FILTER_FACTOR	0.006 to 10.002 PLCs 0 to 100 0 to 100	Settings for the CUSTOM speed setting (see previous modifier)
KI_CVU_AC_SRC_HI KI_CVU_AC_MEAS_LO	1 or 2 (setting HI side to 1 sets LO side to 2, and vice versa)	Use to specify the AC source HI slice and AC source LO side.
KI_CVU_DC_SRC_HI KI_CVU_DC_SRC_LO	1 or 2 (setting HI side to 1 sets LO side to 2, and vice versa)	Use to specify the DC source HI slice and DC source LO side.
KI_CVU_DCV_OFFSET	-30 to +30 (default is 0)	Sets the DC bias offset (in volts).
KI_CVU_MEASURE_MODEL	KI_CVU_TYPE_ZTH KI_CVU_TYPE_RJX KI_CVU_TYPE_CPGP KI_CVU_TYPE_CSRS KI_CVU_TYPE_CPD KI_CVU_TYPE_CSD	Impedance and phase (degrees) Resistance and reactance Parallel capacitance and conductance Series capacitance and resistance Parallel capacitance and dissipation factor Series capacitance and dissipation factor

* Can be set to any floating point number between 0 and 3.0, but will be coerced to 0, 1.5 or 3.

smeasf

Purpose Returns the frequencies used for a sweep.

Format

```
int smeasf(INSTR_ID instid, double *freq_arr);
```

Instrument ID of the Model 4200-CVU: CVU1

freq_arr Returned array of test frequencies

Remarks This function returns the present test frequencies used for a sweep. The frequency values are returned in an array. The frequency values are posted to KITE (Sheet and Graph tabs) after the test has finished.

NOTE *The `smeasfRT` function can instead be used to return sourced sweep frequency values in an array. It will post the frequency values to KITE in real time.*

NOTE *Use the function to return the frequency used for a single measurement.*

smeasfRT

Purpose Returns the sourced frequencies (in real time) for a sweep

Format

```
int smeasfRT(INSTR_ID instid, double *freq_arr, char *colname);
```

`instid`	Instrument ID of the Model 4200-CVU: CVU1
`freq_arr`	Returned array of test frequencies
`colname`	Column name to pass into KITE

Remarks Like the function, the test frequencies for a sweep are returned in an array. However, the frequency values are posted to KITE (Sheet and Graph tabs) in real time. That is, each frequency value appears in the Sheet and Graph tabs after each step of the sweep is executed.

Note that the values are only available in real-time if KITE is running. Otherwise, they are stored in an array in the usual fashion.

The `colname` parameter is used to specify a name (character string) for the data sheet column in KITE. For example, the following function will post the frequency values into the KITE data sheet under a column named freq_arr:

smeasfRT(CVU1, freq_arr, freq_arr);

smeast

Purpose Returns timestamps referenced to sweep measurements or a system timer.

Format

```
int meast(long timerID, double *tarray);
```

`timerID`	ID of the Model 4200-CVU or timer: CVU1 or TIMER1, TIMER2, and so on.
`tarray`	Returned array of timestamps

Remarks This function is used to acquire the timestamp for each measurement step of a sweep. The timestamps are returned in an array. The timestamps are posted to KITE (Sheet and Graph tabs) after the test has finished.

NOTE *The function can instead be used to return timestamps in an array. It will post the frequency values to KITE in real time.*

The timestamp can be referenced to the Model 4200-CVU (`timerID` = CVU) or to a system timer (for example, `timerID` = TIMER1). This function is similar to the function, but is synchronized with a sweep to return a timestamp referenced to each measurement.

NOTE *Use the function to return the timestamp for a single measurement.*

NOTE *LPT maintains a list of measurements to be performed at each sweep point after the forcing instrument has stepped its source (V, I or F). The smeasX and smeasXRT functions simply register the measurement with a master list. If the time measurement precedes the Z measurement, then the wrong timestamp will be returned (the one from the previous measurement).*

Example Programming example #2 acquires a timestamp for each measurement in the sweep.

smeastRT

Purpose Returns timestamps (in real time) referenced to sweep measurements or a system timer.

Format

```
int measRT(long timerID, double *tarray, char *colname);
```

`timerID`	ID of the Model 4200-CVU or timer: CVU1 or TIMER1, TIMER2, and so on.
`tarray`	Returned array of timestamps
`colname`	Column name to pass into KITE

Remarks Like the function, the timestamps are returned in an array. However, the timestamps are posted to KITE (Sheet and Graph tabs) in real time. That is, each timestamp appears in the Sheet and Graph tabs after each measurement is performed.

Note that the values are only available in real-time if KITE is running. Otherwise, they are stored in an array in the usual fashion.

The `colname` parameter is used to specify a name (case-sensitive character string) for the data sheet column in KITE. For example, the following function will post the timestamp values into the KITE data sheet under a column named time_arr:

smeastRT(CVU1, time_array, time_array);

NOTE *LPT maintains a list of measurements to be performed at each sweep point after the forcing instrument has stepped its source (V, I or F). The smeasX and smeasXRT functions simply register the measurement with a master list. If the time measurement precedes the Z measurement, then the wrong timestamp will be returned (the one from the previous measurement).*

smeasv

Purpose Returns the DC bias voltages used for a sweep

Format

```
int smeasv(INSTR_ID instid, double *varray);
```

`instid`	Instrument ID of the Model 4200-CVU: CVU1
`varray`	Returned array of DC bias voltages

Remarks This function returns the DC bias voltages used in a sweep. The values are returned in an array. The voltage values are posted to KITE (Sheet and Graph tabs) after the test has finished.

NOTE *The function can instead be used to return sourced sweep DC bias voltage values in an array. It will post the voltage values to KITE in real time.*

NOTE *Use the function to return the DC bias voltage used for a single measurement.*

smeasvRT

Purpose Returns the sourced DC bias voltages (in real time) for a sweep

Format

```
int smeasvRT(INSTR_ID instid, double *varray, char *colname);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
varray	Returned array of DC bias voltages
colname	Column name to pass into KITE

Remarks Like the function, the sourced DC bias voltages for a sweep are returned in an array. However, the voltage values are posted to KITE (Sheet and Graph tabs) in real time. That is, each voltage value appears in the Sheet and Graph tabs after each step of the sweep is executed.

Note that the values are only available in real-time if KITE is running. Otherwise, they are stored in an array in the usual fashion.

The *colname* parameter is used to specify a name (character string) for the data sheet column in KITE. For example, the following function will post the voltage values into the KITE data sheet under a column named volt_arr:

```
smeasvRT(CVU1, volt_arr, "volt_arr");
```

smeasz

Purpose Performs impedance measurements for a sweep.

Format

```
int smeasz(INSTR_ID instid, long model, long speed, double *result1 double *result2);
```

instid	Instrument ID of the Model 4200-CVU: CVU1
model	Measure model (see Table 15-45).
speed	Speed settings: KI_CVU_SPEED_FAST KI_CVU_SPEED_NORMAL KI_CVU_SPEED_QUIET KI_CVU_SPEED_CUSTOM
result1	Array of the first result of the selected measure model
result2	Array of the second result of the selected measure model

Remarks This function performs an impedance measurement on each step of a voltage or frequency sweep. The measured values for a sweep are returned in arrays. The measured readings are posted to KITE (Sheet and Graph tabs) after the test has finished.

NOTE *The function can instead be used to measure and return sweep impedance measurements. It will post the measured readings to KITE in real time.*

This function uses the same parameter values for `mode` as the `measz` function.

Measurement `speed` settings:

KI_CVU_SPEED_FAST	Performs fast measurements (higher noise)
KI_CVU_SPEED_NORMAL	Selects a balance between speed and low-noise
KI_CVU_SPEED_QUIET	Performs low-noise measurements
KI_CVU_SPEED_CUSTOM	Selects custom settings: Delay factor, filter factor and aperture are set using the `setmode` function.

Before calling `smeasz`, use the `forcev` function to set the DC bias level, the `setfreq` function to set the AC drive frequency and the `setlevel` function to set the AC drive voltage.

NOTE *Use the to measure and return a single impedance measurement.*

Example Programming examples 2 through 5 use the `smeas` function for impedance measurements.

smeaszRT

Purpose Performs and returns (in real time) impedance measurements for a voltage or frequency sweep

Format

```
int smeaszRT(INSTR_ID instid, long model, long speed, double
*result1, char *colname1, double *result2, char *colname2);
```

`instid`	Instrument ID of the Model 4200-CVU: CVU1
`model`	Measure model (see Table 15-45).

`speed`	Speed settings:
	`KI_CVU_FAST`
	`KI_CVU_NORMAL`
	`KI_CVU_QUIET`
	`KI_CVU_CUSTOM`

`result1`	Array of the first result of the selected measure model
`colname1`	Column name to pass into KITE for `result1` array
`result2`	Array of the second result of the selected measure model
`colname2`	Column name to pass into KITE for `result2` array

Remarks Like the `smeasz` function, the measured impedance readings for a sweep are returned in arrays. However, the readings are posted to KITE (Sheet and Graph tabs) in real time. That is, two measurement results appear in the Sheet and Graph tabs after each step of the sweep is executed.

Note that the values are only available in real-time if KITE is running. Otherwise, they are stored in an array in the usual fashion.

The `colname1` and colname2 parameters are used to specify names (character strings) for data sheet columns in KITE. For example, the following function will post the results into the KITE data sheet under columns named result1_arr and result2_arr:

```
smeaszRT(CVU1, 2, KI_NORMAL, result1, "result1", result2
"result2");
```

The other parameters for this function are the same as the ones for the `smeasz` function.

sweepf

Purpose Performs a frequency sweep.

Format

```
int sweepf(INSTR_ID instid, double startf, double stopf,
long *NumPts, double delaytime);
```

`instid`	Instrument ID of the Model 4200-CVU: CVU1
`startf`	Initial frequency for the sweep
`stopf`	Final frequency for the sweep
`NumPts`	Query the number of sweep points
`delayTime`	Delay before each measurement (0 to 999 s)

Remarks This function is used to perform a frequency sweep (see Figure 15-180). The Model 4200-CVU provides test frequencies from 10 kHz to 10 MHz in the following steps:

- 10 kHz to 100 kHz in 10 kHz steps
- 100 kHz to 1 MHz in 100 kHz steps
- 1 MHz to 10 MHz in 1 MHz steps

The frequency points to sweep are set using the `startf` and `stopf` parameters. If an entered value is not a supported frequency, the closest supported frequency will be selected (for example, 15kHz input will select 20 kHz). The sweep can step forward (low frequency to high frequency) or it can step in reverse (high frequency to low frequency).

When the sweep is started, the Model 4200-CVU will step through all the supported frequency points from start to stop. For example, if the start frequency is 800 kHz and the stop frequency is 3 MHz, the CVU will step through the following frequency points:

800 kHz, 900 kHz, 1 MHz, 2 MHz, 3 MHz

The `NumPts` query will return the total number of sweep points. For the above example, `NumPts` will return the value 5.

The `delayTime` parameter sets the Delay that occurs before each measurement. Note that there is an inherent system overhead delay (SD) on each frequency step of the sweep.

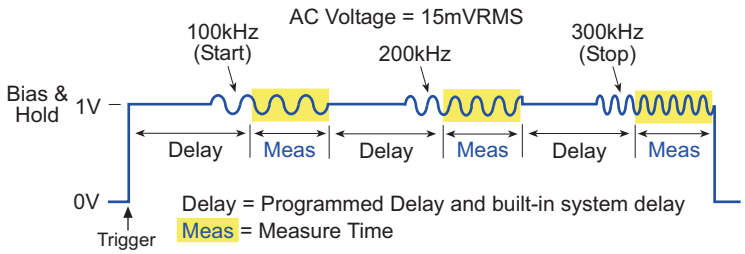
Use the `forcev` function to set the DC bias level and `setlevel` function to set the AC drive voltage.

NOTE *Use the function to perform a dual frequency sweep.*

See also `asweepv`, `dsweepv`, `sweepv`

Example Programming example #3 performs the frequency sweep shown in Figure 15-180.

Figure 15-180
Frequency sweep example



sweepv

Purpose Performs a linear staircase DC voltage sweep.

Format

```
int sweepv(INSTR_ID instid, double startv, double stopv,
long NumSteps, double delaytime);
```

`instid` Instrument ID of the Model 4200-CVU: CVU1

`startv` Initial force value for the sweep (-30 V to 30 V)

`stopv`	Final force value for the sweep (-30 V to 30 V)
`NumSteps`	Number of steps in the sweep (1 to 4096)
`delaytime`	Delay before each measurement (0 to 999 s)

Remarks This function is used to perform a staircase sweep (see Figure 15-181). The linear step size to sweep is set using the `startv`, `stopv` and `NumSteps` parameters. The linear step size for the sweep is calculated as follows:

Step size (in volts) = (`stopv` – `startv`) / `NumSteps`

The sweep can step forward (low voltage to high voltage) or it can step in reverse (high voltage to low voltage).

The `delayTime` parameter sets the Delay that occurs before each measurement. Note that there is an inherent system overhead delay (SD) on each step of the sweep.

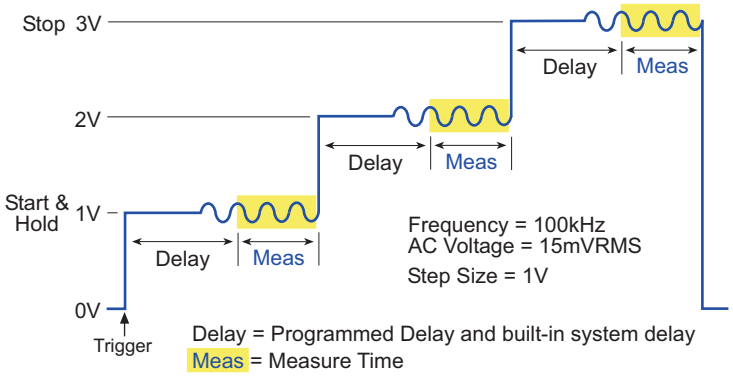
Use the `setfreq` and `setlevel` functions to set the AC drive frequency and voltage for the sweep.

NOTE *Use the function to perform a dual linear staircase voltage sweep.*

See also `asweepv`, `dsweepf`, `sweepf`

Example Programming example #4 performs the voltage sweep shown in Figure 15-181.

Figure 15-181
Voltage sweep example



Programming examples

Programming example #1

Performs a single CsRs impedance measurement. Test parameters:

- DC bias voltage = 1 V
- AC drive frequency = 100 kHz
- AC drive voltage = 15 mV RMS
- Measure model = CsRs

This example also acquires a timestamp for the measurement.

```
double result1, result2, timeStamp;
```

```
int status;
status = forcev(CVU1, 1);          /* Set DC bias to 1 V. */
status = setfreq(CVU1, 100e3);     /* Set AC drive frequency to 100 kHz. */
status = setlevel(CVU1, 15e-3);    /* Set AC drive voltage to 15 mV RMS. */
status = rdelay(0.1);              /* Set settling time to 100 ms. */
status = rangei(CVU1, 1.0e-3);     /* Select 1 mA measure range. */
status = measz(CVU1, KI_CVU_TYPE_CSRS, KI_CVU_SPEED_NORMAL, &result1, &result2);
                                   /* Measure CsRs. */
status = meast(CVU1, &timeStamp); /* Return timestamp for measurement. */
status = devint();                 /* Reset CVU. */
```

NOTE *The* `rdelay` *function is documented in Section 8.*

Programming example #2

Performs a single staircase voltage sweep (see Figure 15-181). CpGp is measured on each step of the sweep. Test parameters:

- AC drive frequency = 100 kHz
- AC drive voltage = 15 mV RMS
- Measure model = CpGp
- Measure range = Auto
- Sweep mode = single
- Start voltage = 1 V
- Stop voltage = 3 V
- Number of steps = 3
- Delay = 50 ms

This example also returns a timestamp for each measurement, and also measures the execution time of the code.

```
double result1[4], result2[4], timeStamp1[4], timeStamp2;
int status;
status = enable(TIMER1);           /* Start timer at 0 seconds. */
status = setfreq(CVU1, 100e3);     /* Set AC drive frequency to 100 kHz. */
status = setlevel(CVU1, 15e-3);    /* Set AC drive voltage to 15 mV RMS. */
status = setauto(CVU1);            /* Select auto measure range. */
status = smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                   /* Configure CpGp measurements. */
status = smeast(CVU1, timeStamp1);/* Return timestamps for all measurements. */
status = rtfary(forceArray);       /* Return array of force voltages. */
status = sweepv(CVU1, 1, 3, 3, 0.05);/* Configure and perform sweep. */
status = meast(TIMER1, &timeStamp2);/* Return execution time for above. */
                                   /* block of code. */
devint();                          /* Reset CVU. */
```

NOTE *The* `enable` *function is documented in Section 8.*

Programming example #3

Performs a single frequency sweep (see Figure 15-180). CpGp is measured on each frequency step of the sweep. Test parameters:

- AC drive voltage = 15 mV RMS
- Measure mode = CpGp
- Measure range = Auto

- Start frequency = 100 kHz
- Stop frequency = 300 kHz
- Number of frequency steps = 3*
- Delay = 50 ms

  * This value is returned from the function to the variable, and not passed by the user.

```
double result1[6], result2[6], forceArray[6];
long NumPts;
int status;
status = setlevel(CVU1, 15e-3);   /* Set AC drive voltage to 15 mV RMS. */
status = setauto(CVU1);           /* Select auto measure range. */
status = smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                  /* Configure CpGp measurements. */
status = rtfary(forceArray);      /* Return array of force frequencies. */
status = dsweepf(CVU1, 100e3, 300e3, &NumPts, 0.05);/* Configure and perform sweep. */
status = devint();                /* Reset CVU. */
```

Programming example #4

Performs a voltage array sweep (see Figure 15-177). CpGp is measured on each point of the sweep. Test parameters:

- AC drive frequency = 100 kHz
- AC drive voltage = 15 mV RMS
- Measure model = CpGp
- Measure range = Auto
- Force array = 1 V, -2 V, 3 V, -4 V
- Number of sweep points = 4
- Delay = 50 ms

```
double result1[4], result2[4], forceArray[4];
int status;
status = setfreq(CVU1, 100e3);    /* Set AC drive frequency to 100 kHz. */
status = setlevel(CVU1, 15e-3);   /* Set AC drive voltage to 15 mV RMS. */
status = forceArray[0] = 1;       /* Force array element 0 = 1 V. */
status = forceArray[1] = -2;      /* Force array element 1 = -2 V. */
status = forceArray[2] = 3;       /* Force array element 2 = 3 V. */
status = forceArray[3] = -4;      /* Force array element 3 = -4 V. */
status = setauto(CVU1);           /* Select auto measure range. */
status = smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                  /* Configure CpGp measurements. */
status = asweepv(CVU1, 4, 0.05, forceArray);/* Configure and perform sweep. */
status = devint();                /* Reset CVU. */
```

Programming example #5

Similar to Programming example #4, but the Delay is set to 0 s, and an array of delay values is used for the sweep (0.5 s, 0.25 s, 0.5 s and 0.25 s).

```
double result1[4], result2[4], forceArray[4], delayArray[4];
int status;
status = setfreq(CVU1, 100e3);    /* Set AC drive frequency to 100 kHz. */
status = setlevel(CVU1, 15e-3);   /* Set AC drive voltage to 15 mV RMS. */
status = forceArray[0] = 1;       /* Force array element 0 = 1 V. */
status = forceArray[1] = -2;      /* Force array element 1 = -2 V. */
status = forceArray[2] = 3;       /* Force array element 2 = 3 V. */
status = forceArray[3] = -4;      /* Force array element 3 = -4 V. */
status = delayArray[0] = 0.5;     /* Delay array element 0 = 0.5 s. */
status = delayArray[1] = 0.25;    /* Delay array element 1 = 0.25 s. */
```

```
status = delayArray[2] = 0.5;     /* Delay array element 2 = 0.5 s. */
status = delayArray[3] = 0.25;    /* Delay array element 3 = 0.25 s. */
status = setauto(CVU1);           /* Select auto measure range. */
status = smeasz(CVU1, KI_CVU_TYPE_CPGP, KI_CVU_SPEED_NORMAL, result1, result2);
                                  /*           Configure CpRp measurements. */
status = adelay(4, delayArray);   /* Call a delay array for asweepv. */
status = asweepv(CVU1, 4, 0, forceArray);/* Configure and perform sweep. */
status = devint();                /* Reset CVU. */
```

Using the Model 4200-CVU card switch matrix

A project plan using a CVU and other instrumentation can be automated using a switching matrix and UTMs to control the switching. When the project plan is run, the switching matrix will automatically make the required instrument connections for each test in the project.

The CVU_ivcvswitch project uses two Model 4200-SMUs and the CVU to provide I-V and C-V testing. The Model 707A or 708 switching mainframe (with Model 7174A matrix card installed) provides the switching. For the I-V test, the switch system connects the SMUs to the DUT (device under test). For the C-V test, the switch system connects the CVU to the DUT.

The following topics provides details on using the CVU with a switch matrix:

- KCON system configuration explains how to add and configure the switch matrix for the Model 4200-SCS system. Keep in mind that the actual CVU connections to the matrix card must match the KCON configuration.
- Connections explains how to connect the CVU to the matrix card.
- After making all the connections between the CVU, matrix card and DUT (wafer), Connection compensation needs to performed in order to achieve optimum measurement accuracy.
- CVU_ivcvswitch is the project plan that uses a switching matrix for an automated system for I-V testing (using SMUs) and C-V testing (using the CVU).

Section 16
Models 4220-PGU, 4225-PMU, and 4225-RPM

In this section:

Supplied accessories

The following items are included with the Models 4220-PGU, 4225-PMU, and 4225-RPM:

Model 4220-PGU. Dual channel pulse generator. Includes the following:

- Model 4220-PGU pulse generator card
- Four SMA-to-SMA cables (male-to-male, 2 m/6.5 ft)
- Two triax-to-SSMC cables
- Two Y-cables

Model 4225-PMU. Dual channel pulse-measure card. Includes the following:

- Model 4225-PMU pulse-measure card
- Four SMA-to-SMA cables (male-to-male, 2 m/6.5 ft)
- Two triax-to-SSMC cables
- Two Y-cables

Model 4225-RPM. Remote pulse and switch module. Includes the following:

- One Model 4225-RPM remote pulse and switch module
- One Model 4200-MAG-BASE (for mounting the 4225-RPM)
- One white RPM communication/measurement cable (2 m/6.5 ft)
- Two SMA-to-SMA cables (male-to-male, 8 in./20.32 cm)
- Two SMA female-to-BNC male connectors
- Two BNC female-to-three-lug triax male connectors
- Four triax protective caps
- Two SMA protective caps

The following items are included with the Model 4200-PMU-PROBER-KIT:

Model 4200-PMU-PROBER-KIT. Includes the following:

- Two BNC male-to-BNC male cables (1.5 m/5 ft)
- Two BNC female-to-SMA male connectors
- Two SMA tees (female-male-female)
- Two SMA-to-SMA cables (male-to-male, 8 in./20.32 cm)
- Two micro-triax-to-SMA (no guard) connectors
- Three BNC female-to-three-lug triax male connectors
- Three BNC female-to-BNC female connectors
- Four SMA female-to-BNC male connectors
- One BNC shorting cap

Models 4220-PGU and 4225-PMU

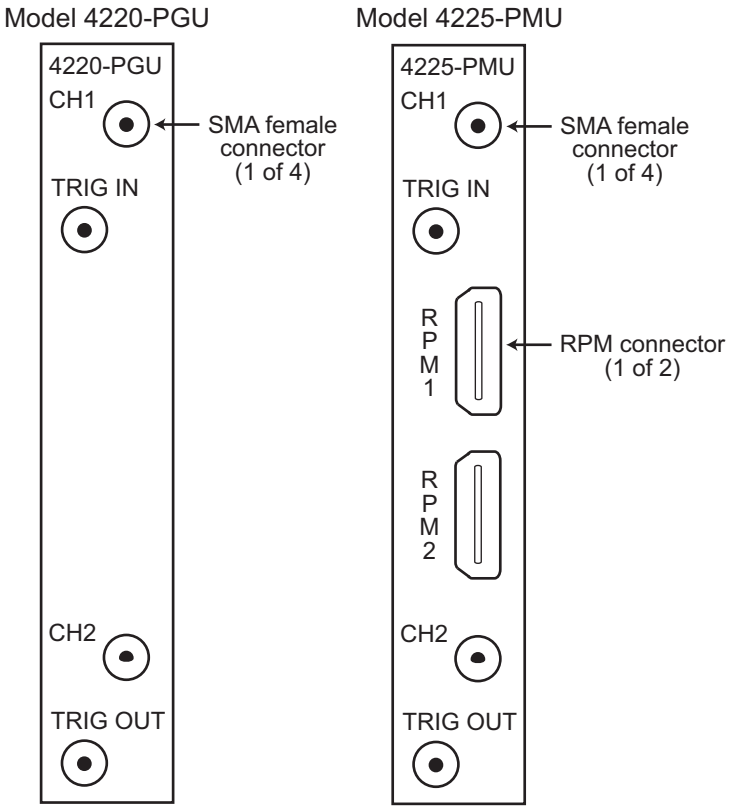
Overview

The Models 4220-PGU and 4225-PMU are high-speed pulse generator cards for the Model 4200-SCS. In this section, the Model 4220-PGU is referred to as a “PGU,” and the Model 4225-PMU is referred to as a “PMU.” The PGU provides pulse output only; the PMU provides both pulse output and pulse measurement.

NOTE *A Model 4225-PMU can be ordered with one or two Model 4225-RPMs. In this section, the Model 4225-RPM is referred to as an "RPM." The RPM is a remote amplifier/switch. It is a preamplifier for the PMU to provide additional low-current measurement ranges, and is also used as a switch for the Models 4225-PMU, 4200-SMU, and 4200-CVU. See Model 4225-RPM for details about the RPM.*

The connectors for the PGU and PMU pulse cards are shown in Figure 16-1.

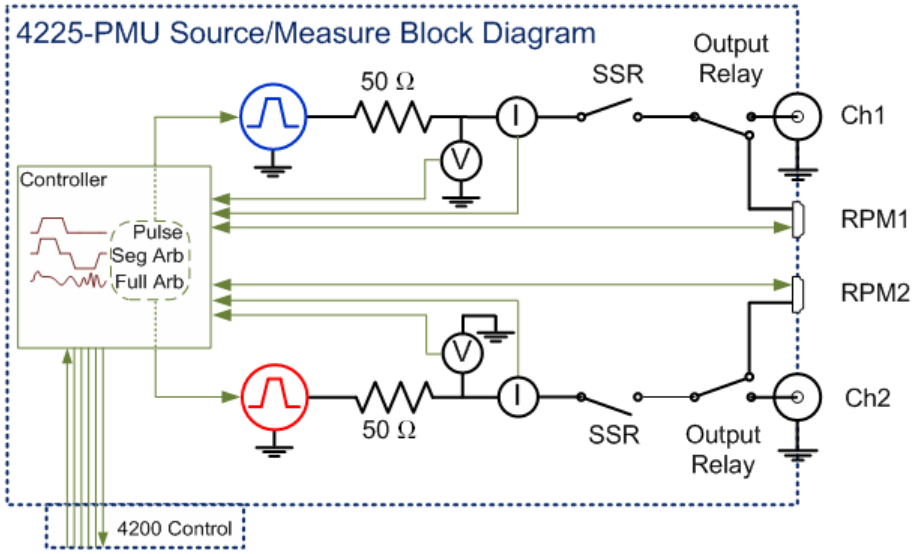
Figure 16-1
Models 4220-PGU and 4225-PMU



NOTE *Use the RPM connectors on the Model 4225-PMU to connect it to the Model 4225-RPM.*

Figure 16-2 shows the block diagram of the PMU. Each channel has two dedicated A/D converters to simultaneously measure current and voltage. The PMU controller controls the two output channels and any RPMs connected to it. The solid-state relays (SSRs) are high-speed and are used to test flash memory. The mechanical output relays are low-leakage. The block diagram for the PGU is similar, except it does not have measure capability and does not have the RPM connectors. Figure 1-21 in the User's Manual shows the simplified schematic of the PGU.

Figure 16-2
Model 4225-PMU block diagram



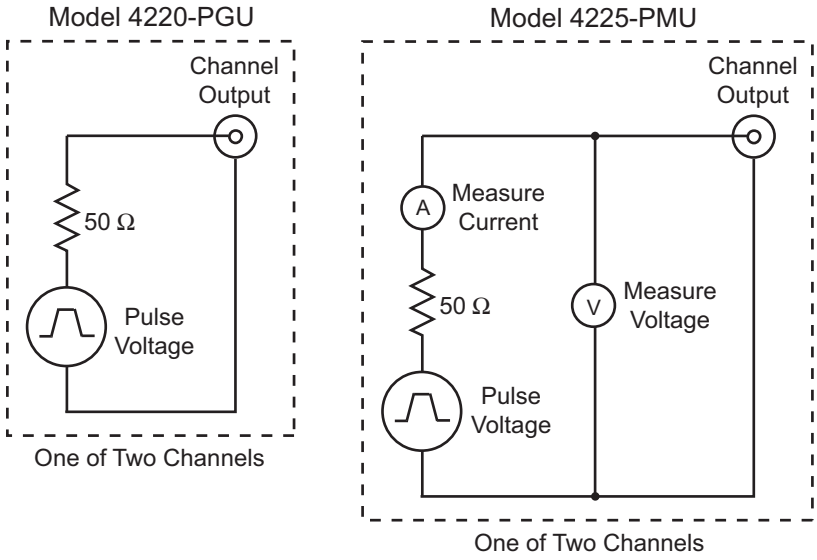
The PGU and PMU are enhanced versions for the Model 4205-PG2. The PGU and PMU have similar pulse output characteristics as the Model 4205-PG2:

- LPT functions: All LPT functions for the Model 4205-PG2 also pertain to the PGU and PMU. Additional LPT functions that pertain to the PGU and PMU are documented in Section 8 (see LPT functions for the Models 4220-PGU and 4225-PMU).
- KPulse: The Keithley Pulse (KPulse) application supports the PG2, PGU (see Section 5 of the User's Manual for details).
- ITMs: The PMU can be configured and controlled using interactive test modules (ITMs). See Procedure to configure a PMU (with or without RPM) from an ITM for details.

NOTE *The pulse source-measure concepts covered in Section 11 apply to the PGU and PMU. However, remember that the PMU has its own measure capability and therefore, you do not need to use the Model 4200-SCP2 or Model 4200-SCP2HR scope card to perform measurements.*

The simplified circuits of the Models 4220-PGU and 4225-PMU pulse generators are shown in Figure 16-3.

Figure 16-3
Simplified circuits of the PGU and PMU



Pulse modes

The PGU and PMU support the following pulse modes:

- Standard pulse mode

  For this two-level pulse mode, the user defines a high and low level for the pulse output. There are two test modes for standard pulse: Pulse IV and waveform capture (see Test modes for details). See Pulse parameter definitions in Section 11 for the standard (two-level) pulse mode. KPulse also supports this pulse mode (see the User's Manual, Standard pulse waveforms, page 5-4).

- Segment ARB pulse mode

  For this multi-level pulse mode, the user defines a pulse waveform that consists of three or more line segments. Segment ARB pulse mode for the PGU and PMU also includes sequencing and sequence looping (see `seg_arb_sequence` and `seg_arb_waveform` in Section 8). Also see Segment ARB waveform in Section 11 for details on parameters. KPulse also supports this pulse mode (see the User's Manual, Segment ARB waveforms, page 5-6.

- Full-arb pulse mode

  For this multi-level pulse mode, the waveform consists of a number of user-defined points (see `arb_array:` and `arb_file:` in Section 8). Also see Full Arb waveform in Section 11 for details on parameters. KPulse also supports this pulse mode (see the User's Manual, Custom file arb waveforms (full-arb), page 5-8).

Standard pulse mode output and timing characteristics

Table 16-1 and Table 16-2 summarize the basic pulse output and timing characteristics of the Models 4220-PGU and 4225-PMU. These output and timing characteristics also apply when using the PMU with an RPM.

Table 16-1
Pulse output characteristics

Output characteristic	Load impedance	High-speed range	High-voltage range
Pulse output	50 Ω	-5 V to +5 V	-20 V to +20 V
	≥1 MΩ	-10 V to +10 V	-40 V to +40 V
Resolution	50 Ω	<250 µV	<750 µV
	≥1 MΩ	<0.5 µV	<1.5 µV
Source impedance	—	50 Ω nominal	50 Ω nominal
Current into 50 Ω load (at full scale)	—	±100 mA	±400 mA
Short circuit current	—	±200 mA	±800 mA

Table 16-2
Pulse timing characteristics

Timing characteristic	High-speed range	High-voltage range
Frequency range	1 Hz to 50 MHz	1 Hz to 10 MHz
Pulse period range	20 ns to 1 s	100 ns to 1 s
Pulse width range	10 ns to (Period - 10 ns)	50 ns to (Period - 10 ns)
Transition time (rise/fall time)	10 ns to 33 ms	50 ns to 33 ms
Timing resolution	10 ns	10 ns

Segment ARB characteristics

Table 16-3 summarizes the basic Segment ARB characteristics of the Models 4220-PGU and 4225-PMU. These characteristics also apply when using the PMU with a Model 4225-RPM.

Table 16-3
Segment ARB characteristics

Segment ARB characteristic	
Maximum number of segments (each channel)	2048
Maximum number of sequences (each channel)	512
Maximum number of sequence loops	10^{12}
Time per segment	20 ns to 40 s
Segment timing resolution	10 ns

Full-arb characteristics

A full-arb waveform can be defined for each pulse card channel using the arb_array (in Section 8) function. A full-arb waveform is made up of user-defined points (up to 262,144 voltage values). A time interval is set to control the time between the waveform points (20 ns to 1 s). A voltage array

is used to set the voltage value for each waveform point. A created full-arb waveform is saved as a `.kaf` file. A full-arb waveform can also be created using KPulse (see the User's Manual, Custom file arb waveforms (full-arb), page 5-8).

Current measurement ranges for the PMU

As shown in Table 16-4, available current measurement ranges for the PMU depend on the selected voltage source range. The 10 mA measure range for the 10 V source range has better accuracy than the 10 mA range for the 40 V source range.

Table 16-4
Fixed current measurement ranges (PMU only)

10 V range	40 V range
10 mA	100 µA
200 mA	10 mA
–	800 mA

The PMU also has autorange and limited autorange.

- Autorange: With autorange selected, the optimum measure range is automatically selected. The available ranges are limited by the chosen pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range.
- Limited autorange: With limited autorange selected, the specified fixed measure range is the lowest range that will be used for automatic ranging. The available ranges are limited by the chosen pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range.

NOTE *When using the PMU with the RPM, additional low-current measurement ranges become available to the PMU (see Current measurement ranges for the PMU with RPM).*

Pulse measurement timing (PMU)

Table 16-5 summarizes typical pulse current measurement timing for the Model 4225-PMU. These times also apply when using the PMU with an RPM.

Table 16-5
Typical current measurement timing

	10 V		40 V		
Current measure range:	**10 mA**	**200 mA**	**100 µA**	**10 mA**	**800 mA**
Recommended minimum pulse width [1]	160 ns	70 ns	6.4 µs	770 ns	770 ns
Recommended minimum measure window	20 ns	20 ns	1 µs	100 ns	100 ns
Recommended minimum transition time (10% to 90%) [2]	20 ns	20 ns	1 µs	100 ns	100 ns
Settling time [3]	100 ns	30 ns	4 µs	500 ns	500 ns

Table 16-5 (continued)
Typical current measurement timing

1. Recommended minimum pulse width = settling time + recommended minimum measure window + recommended minimum transition time.
2. Recommended rise/fall time to minimize overshoot.
3. Time necessary for the signal to settle to the DC accuracy level.

NOTE *When using a Model 4225-RPM with the Model 4225-PMU, additional low-current measure ranges become available. Table 16-8 lists the current measure ranges for the PMU when using an RPM.*

Table 16-6 summarizes typical pulse voltage measurement timing for the Model 4225-PMU.

Table 16-6
Typical voltage measurement timing (PMU only)

Measurement characteristic	High-speed range -10 V to +10 V	High-voltage range -40 V to +40 V
Recommended minimum pulse width [1]	70 ns	150 ns
Recommended minimum measure window	20 ns	20 ns
Recommended minimum transition time (10% to 90%) [2]	20 ns	100 ns
Settling time [3]	30 ns	30 ns

1. Recommended minimum pulse width = settling time + recommended minimum measure window + recommended minimum transition time.
2. Recommended rise/fall time to minimize overshoot.
3. Time necessary for the signal to settle to the DC accuracy level.

Pulse measurement types (PMU)

Table 16-7 summarizes pulse measurement types for the Model 4225-PMU. See Measurement types in Section 8 for detailed information.

Table 16-7
Pulse measurement types

Measurement type	Description
Spot mean discrete	Samples a portion of the high and low levels. The samples are averaged to yield a single current and voltage reading for the high level and low level (see Figure 8-117).
Spot mean average	A specified number of pulse periods are output for the burst sequence. Spot mean discrete measurements are performed for each pulse and then averaged to yield a single voltage and current reading for the high and low levels.
Waveform discrete	The entire pulse is sampled. Sampling is performed on the rise time, top width, and fall time portions of the pulse. A voltage and current reading is yielded for every sample taken on the pulse (see Figure 8-120 and Figure 8-121).

Table 16-7 (continued)
Pulse measurement types

Measurement type	Description
Waveform average	A specific number of pulses are output for the burst sequence. Waveform discrete measurements are performed on each pulse. The corresponding samples for each pulse are then averaged to yield a group of voltage and current readings for the burst sequence.

PMU test modes, measure modes, and sample rate

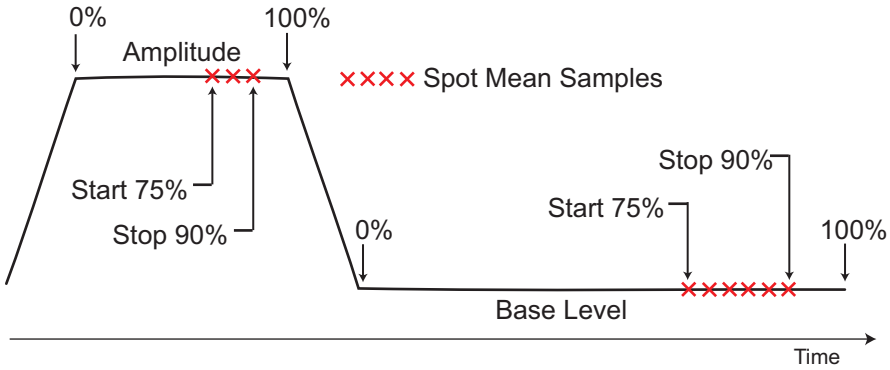
Test modes

There are two source/measure modes for the PMU: Pulse IV and waveform capture.

Pulse IV

For pulse IV, spot mean measurements are performed on pulse amplitude and base level. Voltage and current can be measured. Figure 16-4 shows the measure windows for spot mean measurements. The start measure and stop measure points for interactive test modules (ITMs) are fixed at 75 percent and 90 percent for both amplitude and base level.

Figure 16-4
Pulse IV (spot mean measurements)



The number of measurement samples that are taken is relative to the widths of the magnitude and base level. For example, if the width of the magnitude is 1 µs, the measure window is 750 ns to 900 ns. If the width of the base level is 2 µs, the measure window is 1500 ns to 1800 ns.

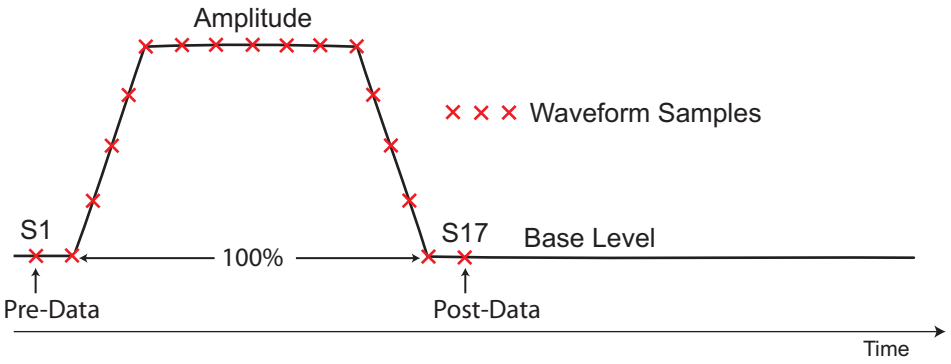
NOTE *Figure 16-132 shows an example of a family of curves generated using the pulse IV test mode.*

NOTE *See Spot mean measurements in Section 8 for details on spot mean (pulse IV) measurements. For user test module (UTM) programming, the pulse_meas_sm and pulse_meas_timing functions are used to configure spot mean measurements. For ITM operation, the measure window is fixed. For UTM operation, the pulse_meas_timing function allows you to adjust the measure window.*

Waveform capture

For waveform capture, measurement samples are acquired on pulse rise, pulse amplitude, pulse fall, and a small portion before the rise (pre-data) and after the fall (post-data) on the base level (see Figure 16-5).

Figure 16-5
Waveform capture



NOTE *Figure 16-121 shows an example of a waveform capture.*

NOTE *See Waveform measurements in Section 8 for details on waveform (capture) measurements. For UTM programming, the pulse_meas_wfm and pulse_meas_timing functions are used to configure waveform measurements. For ITM operation, the pre-data and post-data settings are fixed. For UTM operation, the pulse_meas_timing function allows you to adjust the pre- and post-data settings.*

Measure modes

There are two measure modes for the PMU: Discrete pulses and average pulses.

Discrete pulses

- For pulse IV (spot mean), the averaged voltage and/or current readings for every sampled pulse period are acquired. For ITM operation, the readings are placed in the KITE Sheet tab.
- For waveform capture, enabled voltage and current readings and timestamps for every sample of the waveform are acquired. For ITM operation, the readings are placed in the KITE Sheet tab.

NOTES *See Spot mean discrete readings in Section 8 for details on the discrete pulses measure mode for pulse IV. For UTM programming, the discrete pulse measure mode is called spot mean discrete. The pulse_meas_sm function is used to select the discrete acquisition type.*

See Waveform discrete readings in Section 8 for details on the discrete pulses measure mode for waveform capture. For UTM programming, the discrete pulse measure mode is called waveform discrete. The pulse_meas_wfm function is used to select the discrete acquisition type.

Average pulses

- For pulse IV (spot mean), the mean values of two or more pulses are averaged. Think of it as the “mean of the means.”
- For waveform capture, each acquired reading is a mean average of the corresponding samples for all the pulses in the burst.

NOTES *See Spot mean average readings in Section 8 for details on the average pulses measure mode for pulse IV. For UTM programming, the average pulse measure mode is called spot mean average. The pulse_meas_sm function is used to select the average acquisition type.*

See Waveform average readings in Section 8 for details on the average pulses measure mode for waveform capture. For UTM programming, the average pulses measure mode is called waveform discrete. The pulse_meas_wfm function is used to select the average acquisition type.

Sample rate

For the PMU, the maximum measurement sampling rate for each A/D test is 200e6 (200 million) samples per second. However, there is a limit to the number of samples (one million) that can be acquired per A/D test. When a test is configured to exceed that limit, the sample rate is automatically lowered when using KITE ITMs so that less than one million samples will be acquired.

Pulse IV (spot mean)

The maximum number of samples per A/D per test is <1,000,000 (one million). If an ITM is configured to yield more than one million samples, KITE automatically lowers the sampling rate. For pulse IV (spot mean), typically the sample rate is reduced only if the measure window is wide (due to a wide pulse width or period) or if the number of pulses is large.

The total number of samples for a test is calculated as follows:

Number of samples = Measure window x Sample rate x Number of pulses x Sweep points x Step points

Example: Test that uses a single PMU to perform 50 20-step sweeps

Pulse width = ~7 µs
Measure window = 1 µs
Sample rate = 200e6 samples per second
Number of pulses = 50
Number of steps in sweep = 20

The number of samples acquired for the above example is calculated as follows:

Number of samples = 1 µs x 200e6 x 50 x 20
= 200,000

Because the number of samples is less than the one million sample limit, the sample rate of 200e6 samples per second is used for the above example.

Waveform capture

The maximum number of rows that can appear in the Sheet tab for an ITM is 4096. Each waveform sample uses one row of the sheet. Therefore, the number of samples acquired for a waveform must fit within the 4096 points.

The number of samples (rows) for a waveform that is 20.48 µs wide is calculated as follows:

Number of samples (rows) = Sample rate x waveform width
= 200e6 x 20.48 µs
= 4096

If the waveform was any wider, the sample rate will automatically be lowered by KITE to fit the waveform within the 4096 points.

Waveform width that is sampled includes pulse rise, pulse magnitude, pulse fall, and a small portion of the base level before the rise and after the fall (see Figure 16-5).

Model 4225-RPM

The Model 4225-RPM is a remote amplifier/switch. It is used as a current preamplifier for the Model 4225-PMU. The RPM provides additional high-speed, low-current measurement ranges. Table 16-8 lists the current ranges for the RPM and summarizes the minimum timing characteristics. Note that these suggested minimum timing values do not include settling time of the interconnect or the test device.

The RPM can also be used as a switch for the Models 4200-SMU, 4200-CVU, and 4225-PMU. See Using the RPM as a switch.

Table 16-8
Typical current measurement timing

	10 V					
Current measure range:	**100 nA**	**1 µA**	**10 µA**	**100 µA**	**1 mA**	**10 mA**
Recommended minimum pulse width [1]	134 µs	20.4 µs	8.36 µs	1.04 µs	370 ns	160 ns
Recommended minimum measure window	10 µs	1.64 µs	1 µs	130 ns	40 ns	20 ns
Recommended minimum transition time (10 to 90%) [2]	1 µs	360 ns	360 ns	40 ns	30 ns	20 ns
Settling time [3]	100 µs	15 µs	6 µs	750 ns	250 ns	100 ns

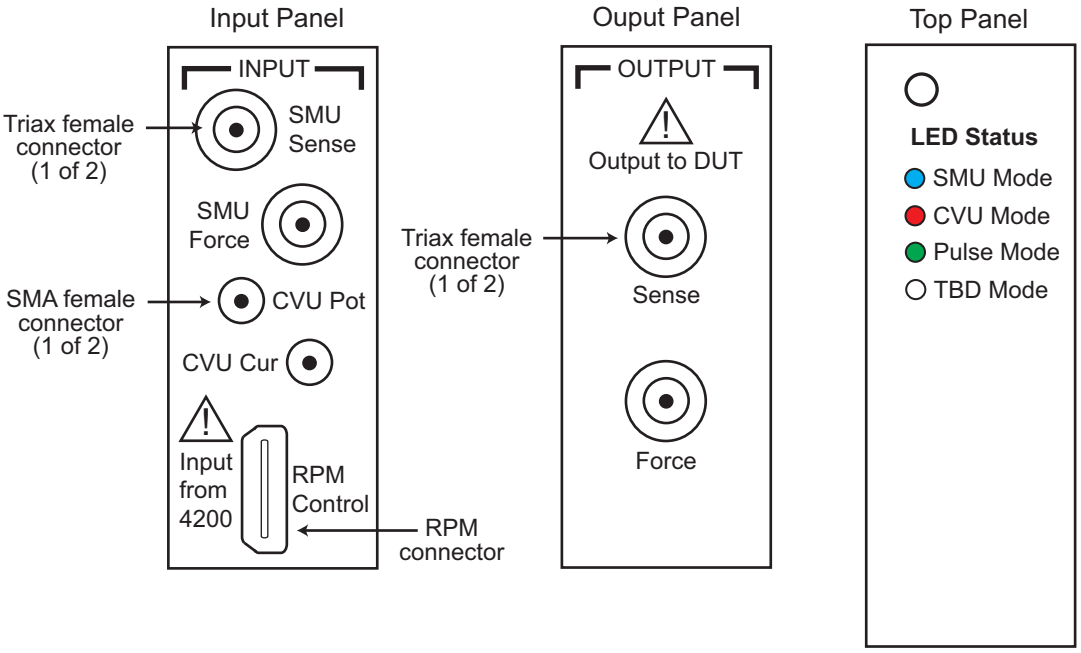
1. Recommended minimum pulse width = Settling time + Recommended minimum measure window + Recommended minimum transition time.
2. Recommended rise/fall time to minimize overshoot.
3. Time necessary for the signal to settle to the DC accuracy level.

RPM input, output, and top panels

The input, output, and top panels for the RPM are shown in Figure 16-6. The RPM connector on the input panel connects to one of the RPM connectors (channel 1 or channel 2) on the Model 4225-PMU. The RPM also has input connectors for a Model 4200-SMU (source-measure unit) and a Model 4200-CVU (C-V unit).

Figure 16-6 shows the modes for the RPM LED colors. Note that the RPM LED shows the mode of the RPM, but not the output status. The output status of the 4200 is indicated by the Active or Measure light on the front of the 4200 chassis. During normal KITE operation, only the red, green or blue colors are shown. However, other colors or color combinations are possible, and are normal part of the RPM operation. During 4225-PMU self-test, the RPM light is green, but there is a portion of the test where the LED flashes red and green. During the 4225-RPM self-test, the RPM LED alternates between purple and green for the majority of the test. During firmware upgrade of the 4225-PMU, the RPM LED is green, but flashes red/green near the end of the process. During firmware upgrade of the 4225-RPM, the RPM LED is blue at the start, changes to green for the remainder of the process.

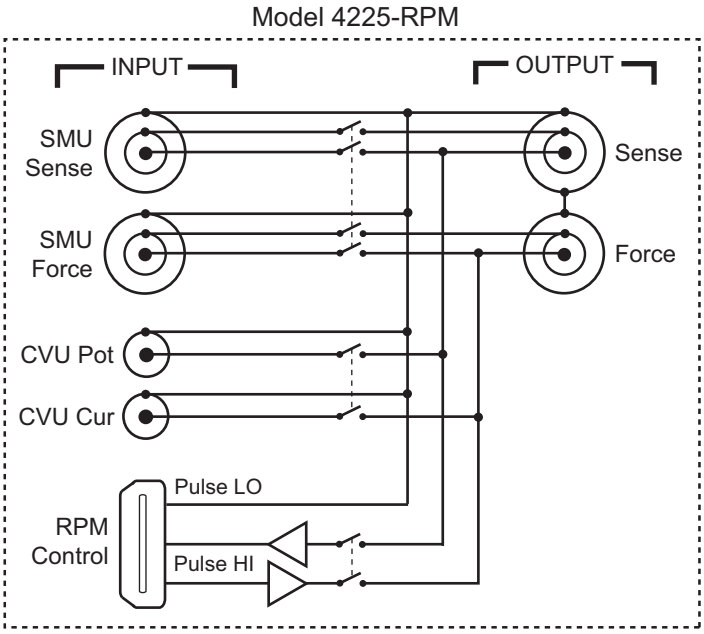
Figure 16-6
Model 4225-RPM



RPM wiring diagram

The internal wiring diagram of the RPM is shown in Figure 16-7.

Figure 16-7
Wiring diagram of the RPM



Signals from the Model 4200-SCS instrument cards are routed through the RPM to the output Force and Sense connectors. Switching is used to control which card is connected to the output. See Using the RPM as a switch for more information on switching.

The LEDs on the top panel (see Figure 16-6) indicate which card is connected to the output. By default, the RPM (pulse mode) is connected to the output unless a SMU or CVU is switched in.

RPM diagrams for local and remote sensing

Figure 16-8 shows the diagram for local sensing. The center conductor of the Force triax connector is connected to the high side of the device under test (DUT) while the outer shield is connected to DUT low. The Sense connector is not used.

Figure 16-8
Diagram for local sensing

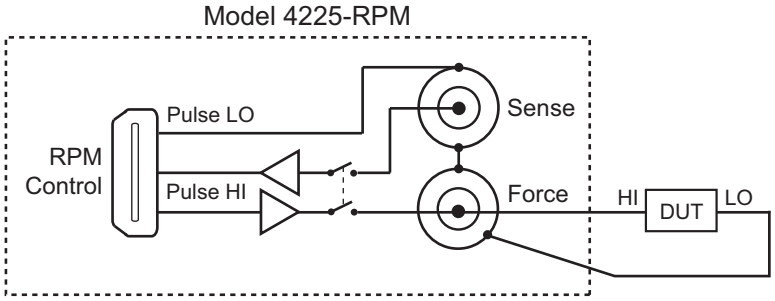
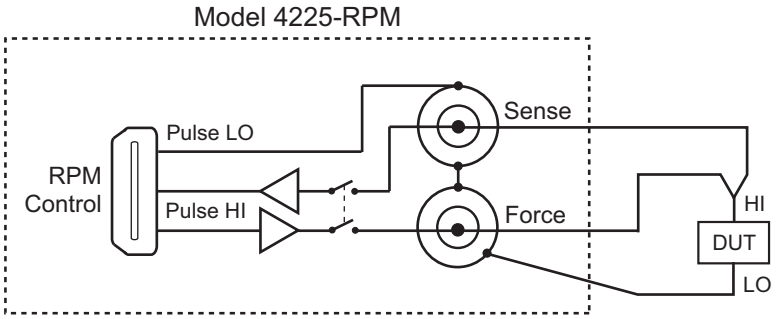


Figure 16-9 shows the diagram for remote sensing. Both Sense and Force are connected to DUT high.

Figure 16-9
Diagram for remote sensing



Current measurement ranges for the PMU with RPM

When using the RPM, the following additional current measurement ranges become available to the PMU: 100 nA, 1 µA, 10 µA, 100 µA, and 1 mA. Table 16-9 lists the current measurement ranges for the PMU when using the RPM. The 10 mA measure range for the 10 V source range has better accuracy than the 10 mA range for the 40 V source range.

Table 16-9
Current measurement ranges for the PMU with RPM

10 V range	40 V range
100 nA	100 µA
1 µA	10 mA
10 µA	800 mA
100 µA	–

Table 16-9 (continued)
Current measurement ranges for the PMU with RPM

10 V range	40 V range
1 mA	–
10 mA	–
200 mA	–

The PMU with RPM also has autorange and limited autorange.

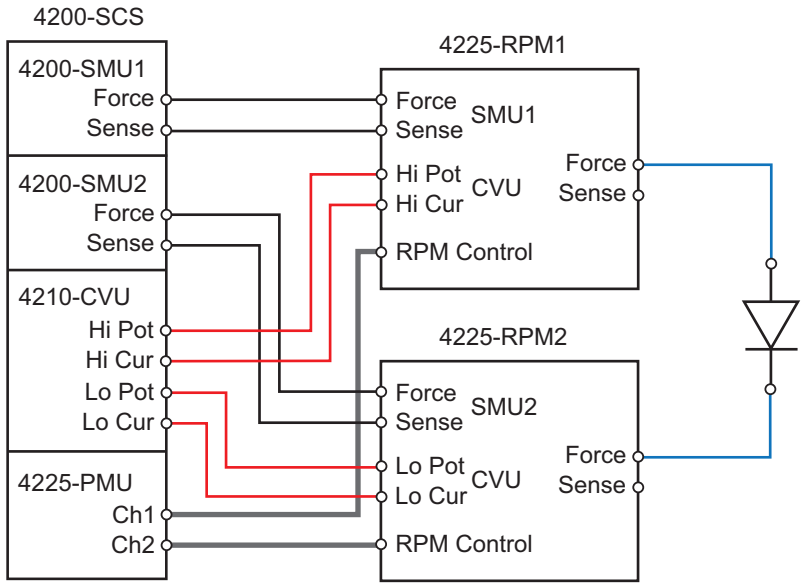
- Autorange: With autorange selected, the optimum measure range is automatically selected for the chosen pulse timing parameters. For shorter pulse widths and pulse periods, the smaller current measure ranges are not available. For additional information, see PMU minimum settling times versus current measure range for the chosen pulse timing parameters.
- Limited autorange: With limited autorange selected, the specified fixed measure range is the lowest range that will be used for automatic ranging.The available ranges are limited by the chosen pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range.

NOTE *When using the PMU without the RPM, the additional low-current measurement ranges are not available (see Current measurement ranges for the PMU).*

Using the RPM as a switch

The RPM can be used to switch a PMU, CVU, or SMU to a DUT terminal. The wiring diagram in Figure 16-7 shows the switches. Figure 16-10 shows a typical test configuration for using an RPM as a switch for a PMU, SMU, and CVU. In general, one RPM per device terminal is recommended. By default, the PMU (with RPM) is connected to the output unless a SMU or CVU is switched in.

Figure 16-10
Test configuration for using RPMs as a switch



Both the red cables (supplied with the CVU) and the blue cables (supplied with the optional Model 4210-MMPC cable kits) are 100 Ω. Remote sensing can be done using the optional Model 4210-MMPC cable kits with the RPMs.

Controlling RPM switching

Before using an RPM, configure the Model 4200-SCS (perform the steps in Update the RPM configuration in KCON). This will properly associate the instruments connected to each RPM. There are two methods to control RPM switching:

- ITM operation using automatic switching (after performing the steps in Update the RPM configuration in KCON)
- Set RPM switching using a UTM for UTM operation

CAUTION **You must Update the RPM configuration in KCON before using the RPM to control switching. If you do not, corrupt test data may result due to incorrect switch settings in the RPM.**

Update the RPM configuration in KCON

The KCON feature Update Preamp and RPM Configuration: must be run:

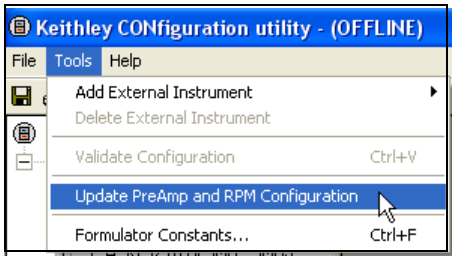
- Whenever an RPM is connected or disconnected from the Model 4200-SCS
- Whenever RPM input connections for a SMU, CVU, or PMU are changed

When using ITMs for DUT testing, you can configure RPM switching in KCON. After you connect the PMU, SMUs, and CVU to the RPM, update the RPM configuration in KCON. When you then open KITE, the system will detect the instrument connections to the RPM. When an ITM is run, KITE will automatically close the appropriate switches to connect the PMU, SMU, or CVU to the output of the RPM. Use the following procedure to configure RPM switching:

Update Preamp and RPM Configuration:

1. Close **KITE** if it is open.
2. On the desktop, double-click the **KCON** icon to open KCON.
3. Make needed connections to the RPM modules in the system (make sure to connect the RPM modules to the correct PMU). To connect or to change connections to the RPM modules, perform the following steps:
   a. Shut down Windows.
   b. Power off the Model 4200-SCS.
   c. Make or break connections as needed between the RPM module and the PMU.
   d. Restart the Model 4200-SCS.
4. At the top of KCON, click **Tools**, and then select **Update PreAmp and RPM Configuration** (see Figure 16-11).
5. Click **File**, and then select **Save**.
6. Exit from KCON.

Figure 16-11
Update RPM configuration



NOTE ***Updating the RPM configuration in KCON does not configure RPM switching for DUT testing using UTMs. You must Set RPM switching using a UTM for UTM operation.***

Set RPM switching using a UTM for UTM operation

For UTM testing from within the user module, use the LPT function rpm_config (this function is defined in Section 8). Make sure you perform the procedure in Update Preamp and RPM Configuration: before you use the LPT command rpm_config to control the RPM.

Connections

The following connection information is provided in this section:

- Connection guidelines
- Basic PMU connection schemes
- Connections to prober or test fixture bulkhead connectors
- Using an SMA to SSMC adapter cable to connect pulse card to DUT
- Model 4225-RPM connections

NOTE *See Supplied accessories for a listing of the cables and connectors included with the PGU, PMU, and RPM.*

There are two optional prober cable kits that are available from Keithley Instruments to provide connections to a DUT:

- Model 4210-MMPC-S: Use this cable kit with the Suss Micro Tec PA200/300 series prober.
- Model 4210-MMPC-C: Use this cable kit with the Cascade Microtech 12000 series prober (manipulator type DCM-200 series).

NOTE *For details on using these prober cable kits, refer to PA-1000 for the Suss prober cable kit and PA-1001 for the Cascade prober cable kit.*

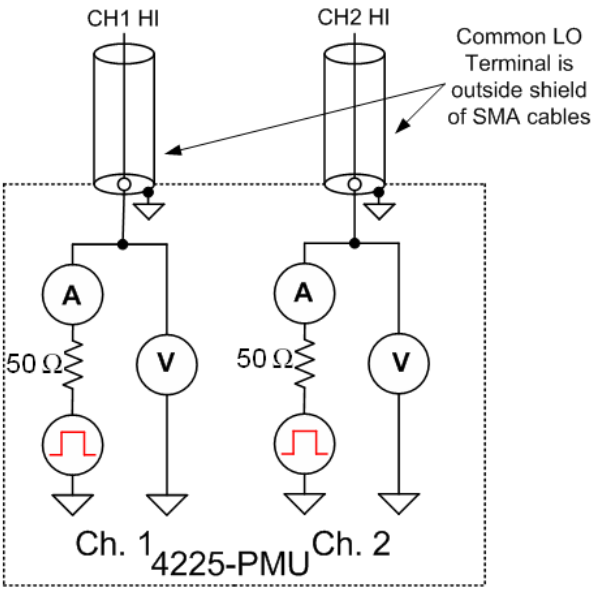
Connection guidelines

Proper connection methods are critical to perform stable and accurate measurements using the PMU (with or without the RPM). It is also a good practice to apply these guidelines to all the Keithley Instruments pulse cards to prevent pulse voltage overshoot and oscillations.

PMU common connections

Common low for the PMU is the outer shells of the two SMA connectors. With an SMA cable connected (see Figure 16-12), common low is the outside shield of the cable.

Figure 16-12
PMU common low terminals



Because pulsing requires high frequency signal propagation, reduce cable inductance by minimizing the loop area of the connection to the device under test (DUT). See Figure 16-13 and Figure 16-14 for a graphic definition of the loop area for the coax cabling.

Because it would create a large loop area, do not use the GNDU as common low for the PMU. When using the GNDU, an inductive loop area is created when the HI and LO leads are separated. Fast rise times (dt), high current (di), and large inductances (L) can cause voltage overshoots, oscillations, and ringing in the high speed measurement circuit. This is based on Lenz's law: V = L di/dt.

Shield connections

For multiple PMU channels, you should connect the shields (common low) from all PMU channels as close as possible to the DUT. You will reduce inductance by minimizing the loop area of the shield connections. The Basic PMU connection schemes show how the shields should be connected. Figure 16-18, Figure 16-20, and Figure 16-21 illustrate proper shield connection schemes using the supplied cabling.

Cable length

Use the shortest possible cable length to achieve the highest frequency output, the best pulse shape, and the best results. Here are reasons to avoid using longer cable lengths:

- Longer cable lengths have longer reflection times, which can slow down transmission times.
- Longer cables may have impedance mismatches, which can cause distortions.
- Higher capacitance in longer cables causes higher capacitive charging effects during the pulse transitions (see PMU capacitive charging/discharging effects).

Only use the white SMA coaxial cables that are supplied with the PMU and RPM. These are 50 Ω cables that match the internal 50 Ω resistance of the PMU. The PMU is supplied with 6.5 ft (2 m)

SMA cables and the RPM is supplied with 8-inch SMA cables. Always use the short 8 in. (20 cm) SMA cables with the RPM.

High frequency connections

Use these connection guidelines for high-speed testing (pulse width <1 µs).

- Use cables and connectors optimized for high frequency (at least 150 MHz). The SMA coaxial cables supplied with the PMU and RPM are rated for high frequency.
- Probe manipulators must be rated at least 150 MHz.
- Properly connect the shields of the coaxial cables and minimize the loop area of the shield connections (see Shield connections).
- Minimize cable length (see Cable length).
- Use a signal path that matches the impedance of the instrument (50 Ω). The SMA cables supplied with the PMU and RPM are 50 Ω.

Prober chuck connections

When possible, avoid pulse connections to the prober chuck. If unavoidable, use these guidelines when connecting to the prober chuck:

- When making connections to the back side of the wafer, PMU functionality will be diminished. Use caution and verify waveforms.
- Generally, the chuck adds capacitance and noise. This reduces both low current and high speed sampling performance.
- If one of the device terminals is the back side of the wafer, then pulse only on that terminal (on chuck) and measure at another terminal using the second channel. If possible, do not measure from the PMU channel connected to the chuck.
- For a two-terminal device, use Figure 16-14 as a guide; for a four-terminal device, use Figure 16-16 or Figure 16-21 as a guide (as applicable). This cabling approach permits the low-side measurement approach described in PMU capacitive charging/discharging effects.

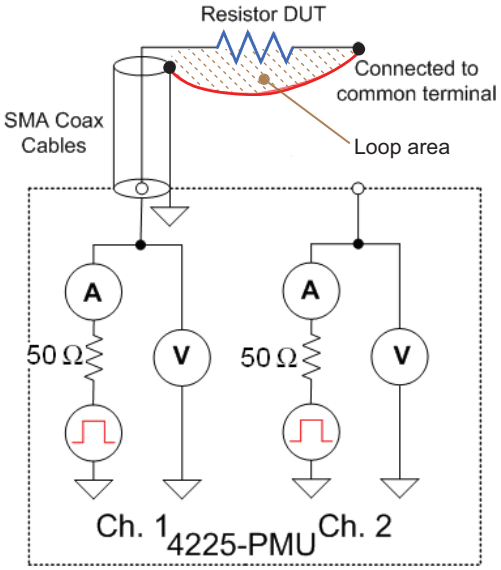
Basic PMU connection schemes

Common lows for a pulse card are the outer shells of the two SMA connectors. With an SMA cable connected (see Figure 16-12), common low is the outside shield of the cable. The connection schemes include the simplified schematic of the PMU.

Two-terminal device connections

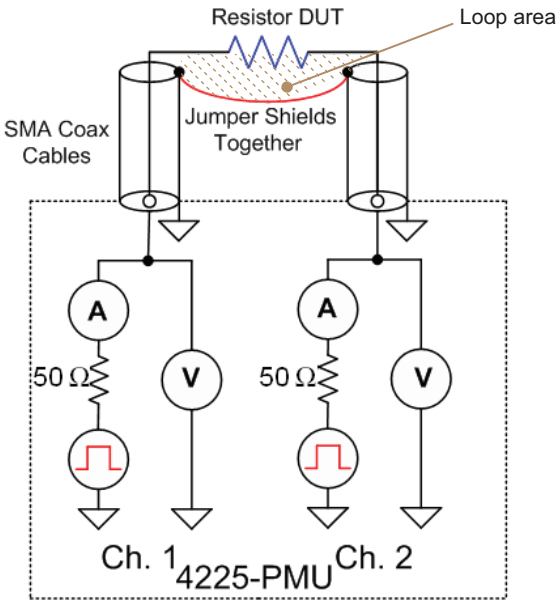
Figure 16-13 shows connections to a two-terminal device using a single channel. Connect one end of the device to the center conductor of Ch 1 and connect the other side to pulse card common (outside shield of the SMA cable).

Figure 16-13
Two-terminal device connections to a pulse card using one channel



You can also connect a two-terminal device to the two channels of a PMU, as shown in Figure 16-14. In this case, channel 1 will source/pulse voltage, and channel 2 will measure the resulting current. Make sure you connect the shields of the SMA cables close to the device under test. This method avoids problems of capacitive charging (see PMU capacitive charging/ discharging effects).

Figure 16-14
Two-terminal device connections to a PMU using both channels

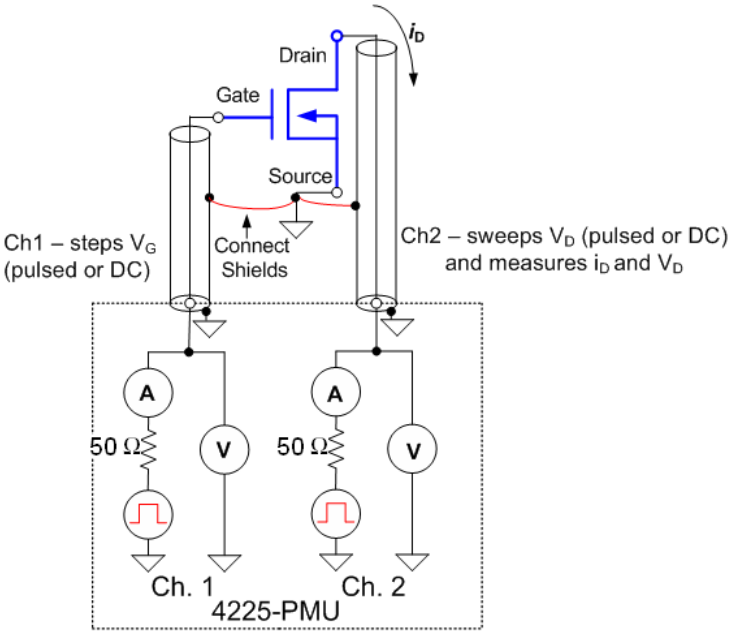


Three-terminal device connections

A three-terminal device can be connected using either two or three PMU channels, depending on if source and or measuring must be performed at each device pin. An example of both channels of a single PMU connected to a three-terminal MOSFET is shown in Figure 16-15. In this example,

connect the Gate terminal to channel 1 of the PMU and connect the Drain terminal to channel 2. Connect the source terminal to the outside shield of channel 2.

Figure 16-15
Three-terminal device connections to a PMU using both channels

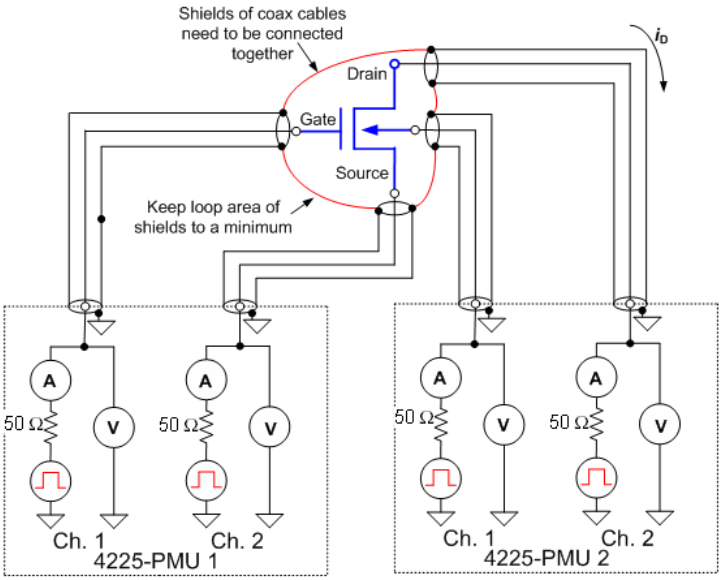


If ultra fast I-V sourcing and measuring is required at each device terminal, then a second PMU is required for the source terminal. Up to four PMUs (eight channels) can be installed in one Model 4200-SCS mainframe.

Four-terminal device connections

To test a four-terminal device, two PMUs are usually required. Figure 16-16 shows the four PMU channels connected to a four-terminal MOSFET. This configuration enables the user to have complete flexibility to enable pulsing and measuring at any terminal on the device. Notice that the shields of the SMA cables from all four channels are connected as close as possible to the device under test.

Figure 16-16
Four-terminal device connections to two PMUs



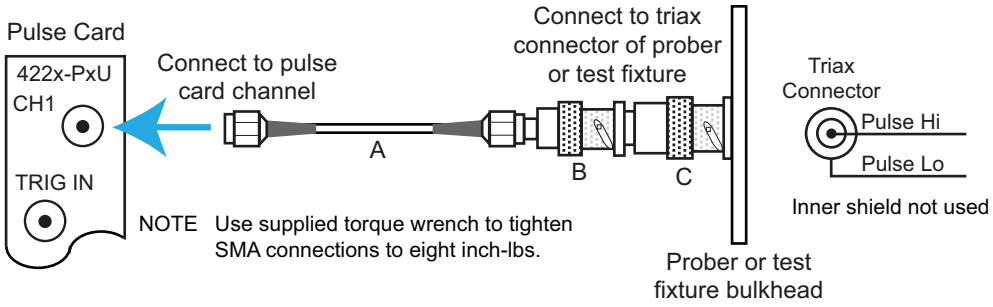
Connections to prober or test fixture bulkhead connectors

The Model 4200-PMU-Prober-Kit (available from Keithley Instruments) is a collection of standard and custom connectors and accessories used to connect the pulse generator to a common variety of prober stations. This kit can also be used for pulse card connections to a test fixture.

Figure 16-17 shows an example of how a PMU or PGU can be connected to a triax connector of a prober or test fixture.

NOTE *If connecting to a prober or test fixture that uses BNC connectors, adapter C is not used.*

Figure 16-17
Pulse card connections to triax prober or test fixture

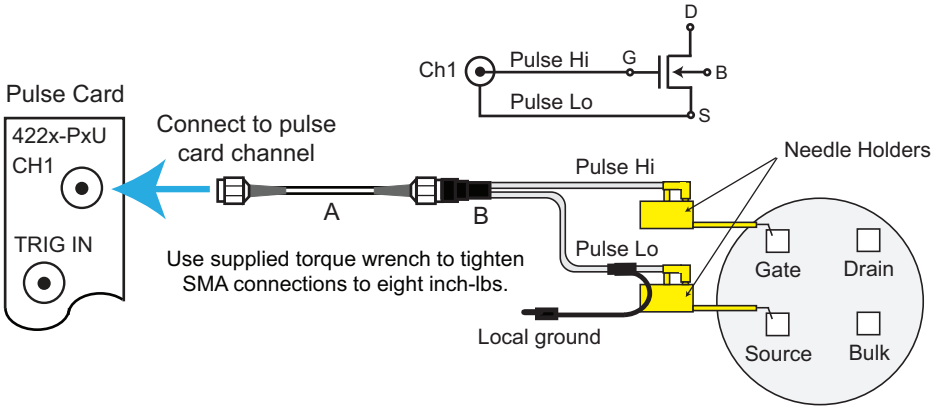


A White SMA cable (2 m/6.5 ft, included with the PGU and PMU)
B SMA female-to-BNC male (included with the Model 4200-Prober-Kit)
C BNC female-to-three-lug triax male (included with the Model 4200-Prober-Kit))

Using an SMA to SSMC adapter cable to connect pulse card to DUT

Figure 16-18 shows an example of how to make pulse card connections to the device under test (DUT) using the supplied adapter cable. For further information on the SMA to SSMC adapter cable, see the 4200-PRB-C link on the Application Note page of the 4200 Complete Reference.

Figure 16-18
Pulse card connections using the SMA to dual SSMC adapter cable



A White SMA cable (2 m/6.5 ft, included with the PGU and PMU)
B SMA to SSMC Adapter Cable (4200-PRB-C, included with the PGU and PMU)

The needle holders shown in Figure 16-18 are supplied by the user.

Model 4225-RPM connections

RPM connection to the PMU

NOTE *The RPM is matched to a PMU card and channel. Make sure to connect the RPM only to that PMU card and channel.*

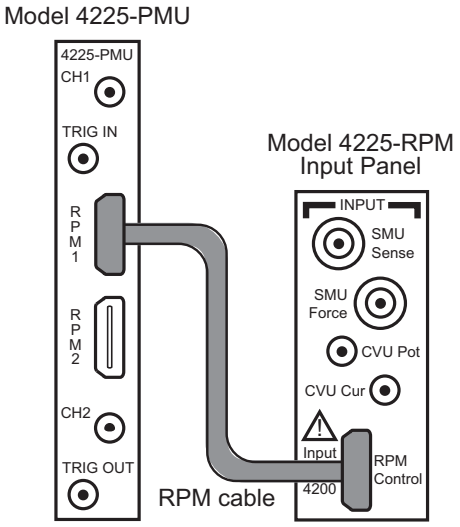
CAUTION **Turn off system power before connecting or disconnecting the RPM to or from the PMU. Failure to do so may result in PMU or RPM damage, possibly voiding the warranty.**

CAUTION **With an RPM installed, NEVER make connections directly to any of the SMA connectors (CH1 and CH2) on the PMU module; this may result in damage to the PMU or DUT, or may produce corrupt data.**

With system power off, use the supplied RPM cable to connect a Model 4225-RPM to the matching RPM channel of the Model 4225-PMU (see Figure 16-19).

NOTE *After connecting or removing an RPM, always perform the Update Preamp and RPM Configuration: procedure to ensure that KCON accurately represents the present Model 4200-SCS hardware configuration.*

Figure 16-19
PMU connection to the RPM



NOTE *If you want to use the PMU without the RPM, you must update the RPM configuration in KCON (see Update Preamp and RPM Configuration:).*

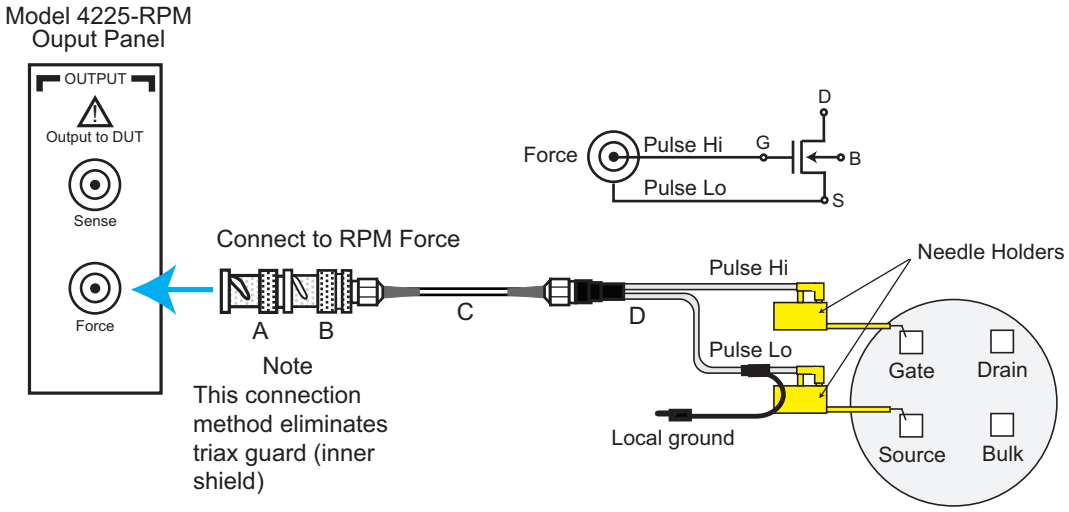
RPM connections to DUT

A device under test (DUT) can be tested using local sensing or remote sensing. Local sensing is performed at the RPM, while remote sensing is performed at the DUT. When using remote sensing, errors due to voltage drops in the Force path between the RPM and the DUT are eliminated. With proper cabling, SMU or CVU tests provide remote sensing through the RPM.

Local sensing

For local sensing, only the Force output terminal of the RPM is connected to the DUT. The Sense output terminal is not used. Figure 16-20 shows local sense connections using the supplied adapter cable and adapters. For the two-terminal test shown in Figure 16-20, the local ground is left unconnected. Test circuit low is connected to the shield of the Force connector through the cables. For further information on the SMA to SSMC adapter cable, see the 4200-PRB-C link on the Application Note page of the 4200 Complete Reference.

Figure 16-20
Two-terminal local sense connections using the SMA to dual SSMC adapter cable



A Triax male-to-BNC female adapter (supplied with the RPM)
B BNC male-to-SMA female adapter (supplied with the RPM)
C White SMA cable (8 in./20.32 cm, supplied with the RPM)
D SMA to SSMC Adapter Cable (4200-PRB-C, supplied with the PMU)

The needle holders shown in Figure 16-20 and Figure 16-21 are supplied by the user.

NOTE *When using two RPMs for four-terminal testing, two Y-cable assemblies are required. Make sure to connect the two local grounds of the two cable assemblies together (see Figure 16-21).*

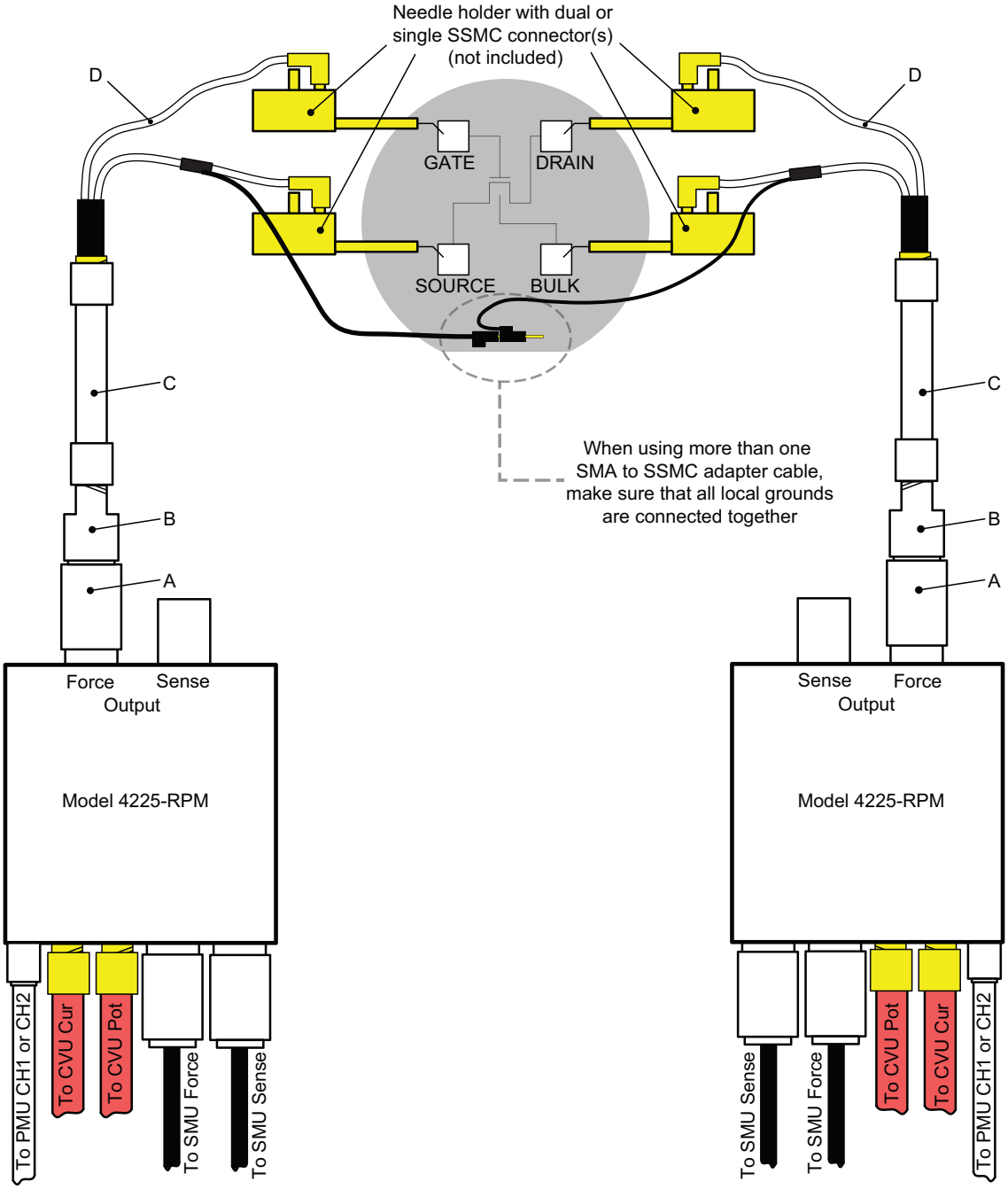
Remote sensing

Optional prober cable kits are available from Keithley Instruments. These kits provide remote sense connections to a DUT:

- Model 4210-MMPC-S: Use this cable kit with the Suss Micro Tec PA200/300 series prober.
- Model 4210-MMPC-C: Use this cable kit with the Cascade Microtech 12000 series prober (manipulator type DCM-200 series).
- Model 4210-MMPC-L: Use this cable kit with a Lucas Signatone Wavelink series prober.
- Model 4210-MMPC-W: Use this cable kit with the Wentworth prober.

NOTE *For details on using these prober cable kits, refer to PA-1000 for the Suss prober, PA-1001 for the Cascade prober, PA-1080 for the Lucas Signatone prober, and PA-1085 for the Wentworth prober.*

Figure 16-21
Four-terminal local sense connections using the SMA to dual SSMC adapter cable



A Triax male-to-BNC female adapter (supplied with the RPM)
B BNC male-to-SMA female adapter (supplied with the RPM)
C White SMA cable (8 in./20.32 cm, supplied with the RPM)
D SMA to SSMC Adapter Cable (4200-PRB-C, supplied with the PMU)

Configuring the PGU, PMU, and RPM using UTMs and ITMs

Configuring the PGU

You must use user test modules (UTMs) to configure and control the PGU. For UTM programming, see LPT functions for the Models 4220-PGU and 4225-PMU in Section 8.

Configuring the PMU and RPM preamp

You can use interactive test modules (ITMs) to configure and control a PMU and RPM preamplifier. Procedure to configure a PMU (with or without RPM) from an ITM provides the steps to configure an ITM for the PMU.

UTMs (user test modules) can also be used to configure and control the PMU and RPM preamplifier. For UTM programming, see LPT functions for the Models 4220-PGU and 4225-PMU in Section 8.

Configuring an RPM switch

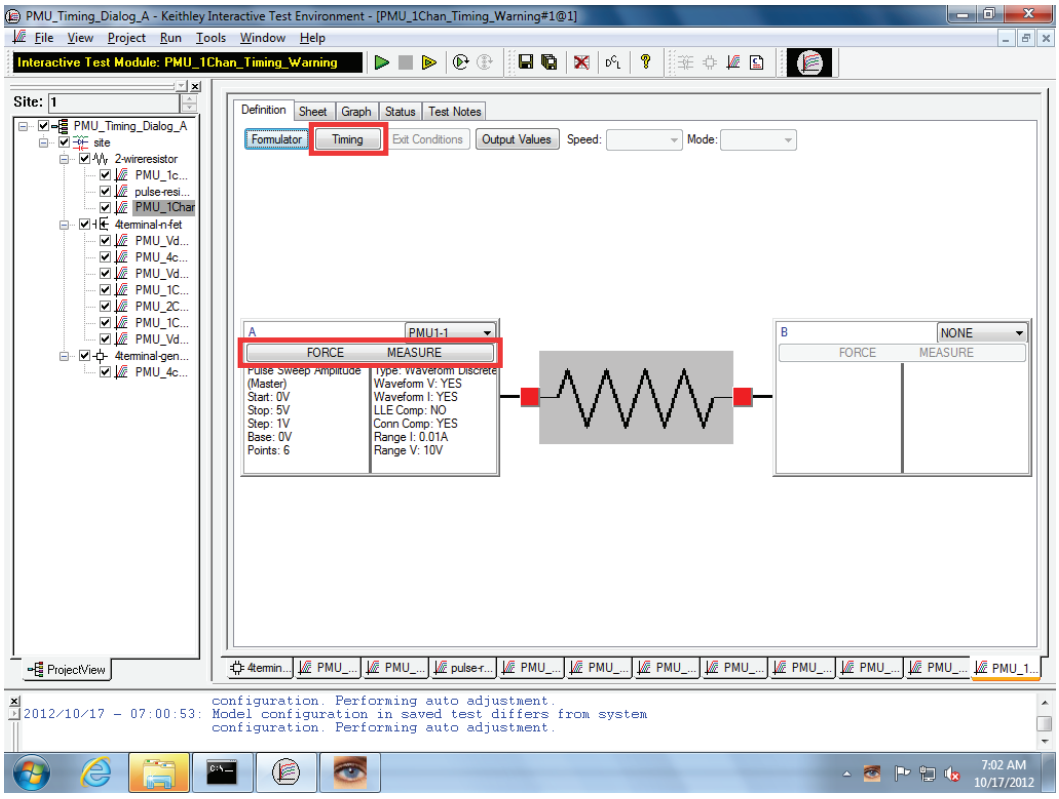
RPM switching can only be controlled using the rpm_config LPT function in a UTM. An ITM automatically controls the RPM based on the type of test (pulse, CV, SMU) (see Using the RPM as a switch for details).

NOTE *For automatic switching of the RPM in ITMs, the RPM must already have all instrument cabling connected and recognized by the system. Information on performing this update is contained in "Tools > Update DC Preamp and RPM Configuration" on page 7-9.*

Procedure to configure a PMU (with or without RPM) from an ITM

From the KITE interface, open a project/test that uses a PMU. Figure 16-22 shows the Definition tab for an ITM that uses a PMU to test a resistor.

Figure 16-22
ITM Definition tab



Perform the following steps to configure the PMU (with or without the RPM) from an ITM. This step order is recommended, but not required.

Step 1) Select the test and measure modes

Step 2) Select the forcing function

Step 3) Set the measure ranges

Step 4) Configure the selected forcing function

Step 5) Configure measurements

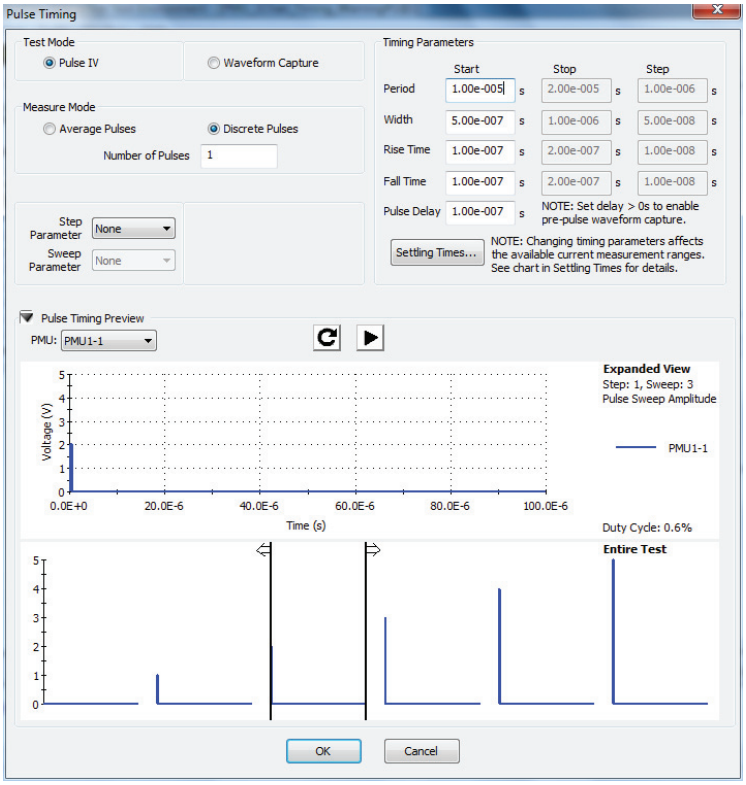
Step 6) Configure pulse timing

Step 1) Select the test and measure modes

In the Definition tab (shown in Figure 16-22), click the **Timing** button to open the Pulse Timing Dialog window (shown in Figure 16-23).

NOTE *The PMU ITM only supports the standard 2-level pulse output mode.*

Figure 16-23
Test and measure modes



In the timing window, select the Test Mode (**Pulse IV** or **Waveform Capture**), Measure Mode (**Average Pulses** or **Discrete Pulses**), and the number of pulses to output. When finished, click **OK** to close the window.

Test Mode

For the Test Mode, select either Pulse IV using spot means (Average Pulses) or Waveform Capture using samples (Discrete Pulses).

- **Pulse IV**: This test mode performs spot mean measurements (V and/or I) of the amplitude and/or base portions of one or more pulses. Typically, current versus voltage is graphed in the Graph tab.
  Figure 16-132 shows an example of a family of curves generated using the pulse IV test mode.
- **Waveform Capture**: This test mode samples the entire waveform. With time stamps enabled, the waveform is graphed (voltage and/or current versus time) in the Graph tab.
  Figure 16-121 shows an example of a waveform capture.

NOTE *See Test modes for more information.*

Measure Mode

For Measure Mode, select either Average Pulses or Discrete Pulses which are both available for Pulse IV or Waveform capture.

- **Average Pulses**: For Pulse IV, the mean values of two or more pulses are averaged. Think of it as the “mean of the means” (see Pulse IV (Average pulses) measurement example). For Waveform Capture, Averaging Pulses means that each sample in the waveform is

averaged with the same point in subsequent waveforms, for each pulse specified by the number of pulses.

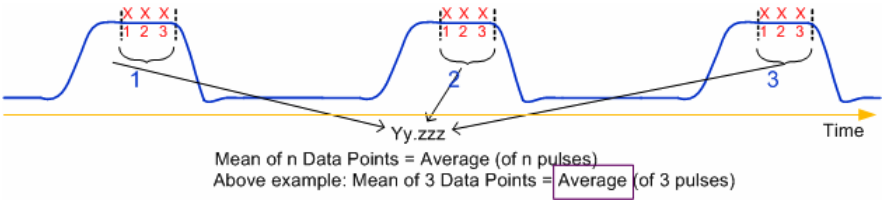
- **Discrete Pulses**: For Pulse IV, the mean of each pulse is acquired (see Pulse IV (Discrete pulses) measurement example). For Waveform Capture, each sample for every waveform is acquired and put in the Sheet tab (see Waveform Capture (Discrete Pulses) measurement example).
- **Number of Pulses**: This is the number of pulses for the PMU channel to output and measure (1 to 10,000) for each step in the sweep. If autoranging, LLEC, or Thresholds are enabled, then the number of pulses will be output multiple times for each step in the sweep.

NOTE *See Measure Mode for additional information on measure modes.*

Pulse IV (Average pulses) measurement example

For the example shown in Figure 16-24, the mean of three pulses are averaged into a single reading (also called a spot mean). One averaged reading is yielded for each pulse. The result of the three averaged readings is placed in the Sheet tab.

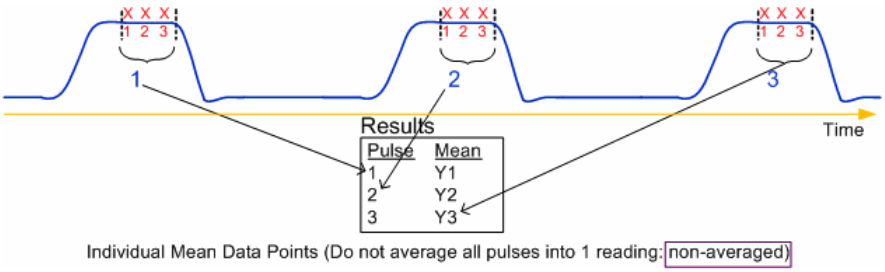
Figure 16-24
Pulse IV - Average pulses

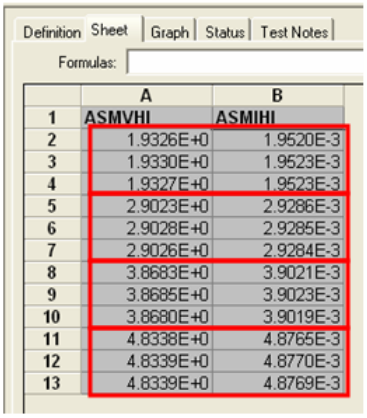


Pulse IV (Discrete pulses) measurement example

For the example shown in Figure 16-25, the readings are the result of a pulsed IV sweep from 2 V to 5 V (in 1 V steps) with the discrete Number of Pulses set to three. This test yields the spot mean of the three pulses for each step of the sweep. The Sheet tab for the 12 readings are also included in Figure 16-25.

Figure 16-25
Pulse IV - Discrete pulses



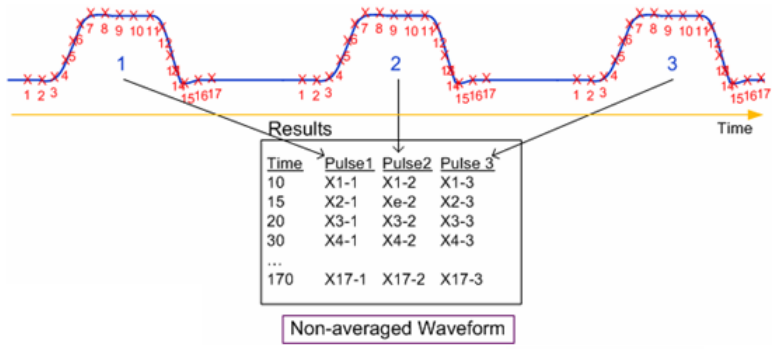


	A	B
1	ASMVHI	ASMIHI
2	1.9326E+0	1.9520E-3
3	1.9330E+0	1.9523E-3
4	1.9327E+0	1.9523E-3
5	2.9023E+0	2.9286E-3
6	2.9028E+0	2.9285E-3
7	2.9026E+0	2.9284E-3
8	3.8683E+0	3.9021E-3
9	3.8685E+0	3.9023E-3
10	3.8680E+0	3.9019E-3
11	4.8338E+0	4.8765E-3
12	4.8339E+0	4.8770E-3
13	4.8339E+0	4.8769E-3

Waveform Capture (Discrete Pulses) measurement example

For the example shown in Figure 16-26, the samples of three pulses are captured. The 52 samples (17 samples x 3 pulses) are placed in the Sheet tab.

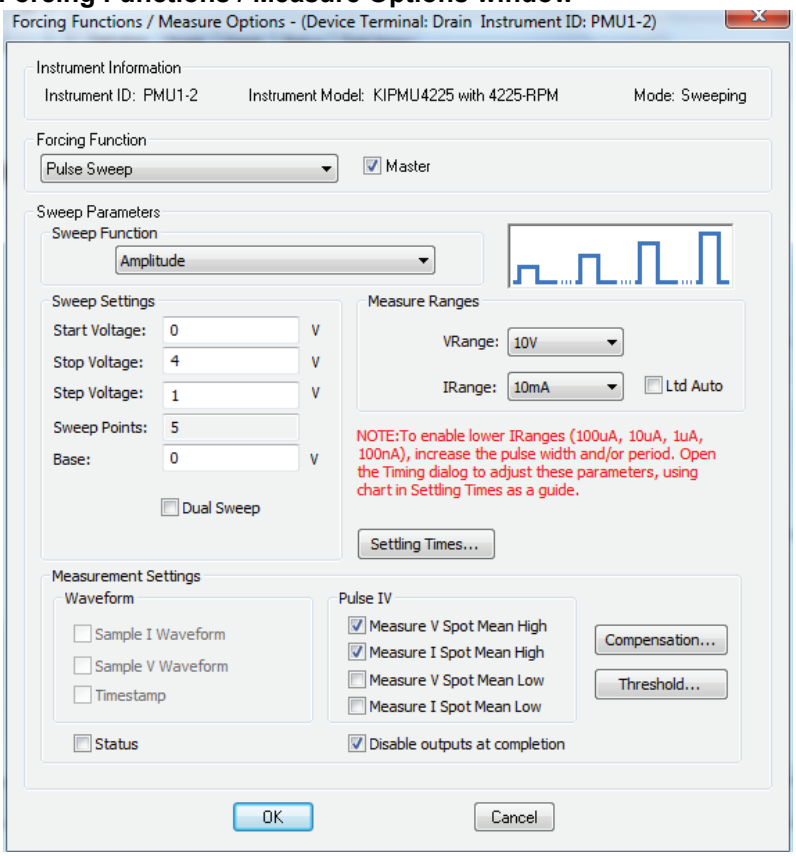
Figure 16-26
Waveform capture - Discrete pulses



Step 2) Select the forcing function

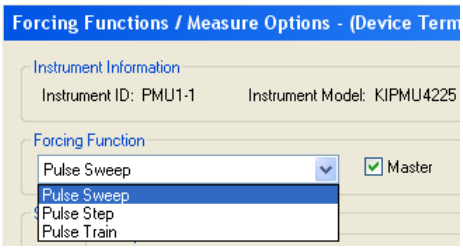
In the Definition tab (shown in Figure 16-22), click the **FORCE MEASURE** button to open the Forcing Functions / Measure Options window (see Figure 16-27).

Figure 16-27
Forcing Functions / Measure Options window



In the Forcing Functions / Measure Options window, select the Forcing Function (**Pulse Sweep**, **Pulse Step**, or **Pulse Train)** from the drop-down menu (see Figure 16-28).

Figure 16-28
Forcing Function



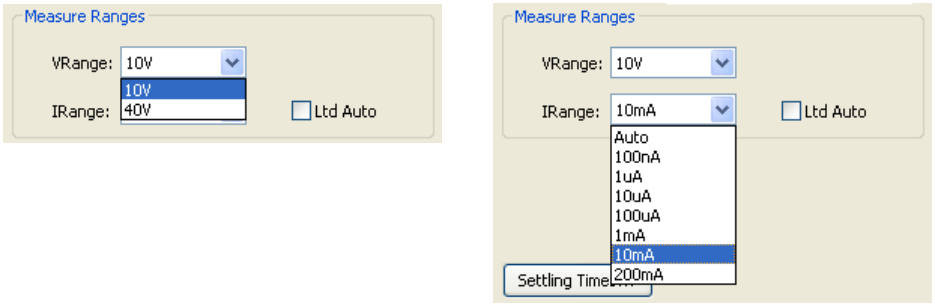
A **Pulse Sweep** and **Pulse Step** function are identical. However, in a test where two or more PMUs are used, the sweep for one PMU is performed on each step of the pulse step of the other PMU. A pulse step requires that at least one channel be configured for Pulse Sweep or Pulse Train. A **Pulse Train** simply outputs one or more pulses of the same magnitude and base level.

When there are two or more PMUs in the test system, the Master check box is used to select one of them as the master PMU. When there is only one PMU in the system, it does not matter whether or not the box is checked.

Step 3) Set the measure ranges

In the Forcing Functions / Measure Options window, set the voltage and current measure ranges from the drop-down menus (see Figure 16-29).

Figure 16-29
Voltage and current measurement ranges



Use the VRange menu to select the voltage source range for the PMU channel. It is not possible to change the voltage range during the test. Note that the 40 V range has a higher maximum output voltage and current, but does not have the faster transition times of the 10 V range or the lower current measure ranges of the 4225-RPM.

Use the IRange menu to select the current measure range for the PMU channel. With Auto selected, the optimum current measure range is automatically selected. With the Ltd Auto (limited auto) box checked, the selected measure range is the lowest range that will be used for automatic ranging. The available ranges are limited by the chosen pulse timing parameters. For additional information, see PMU minimum settling times versus current measure range.

The current measure range affects the time required to obtain settled measurements. Lower current ranges require additional time to reach a settled signal level. If the pulse timing parameters are too short for one or more current measure ranges, the note above the Settling Times will be red and list the unavailable ranges.

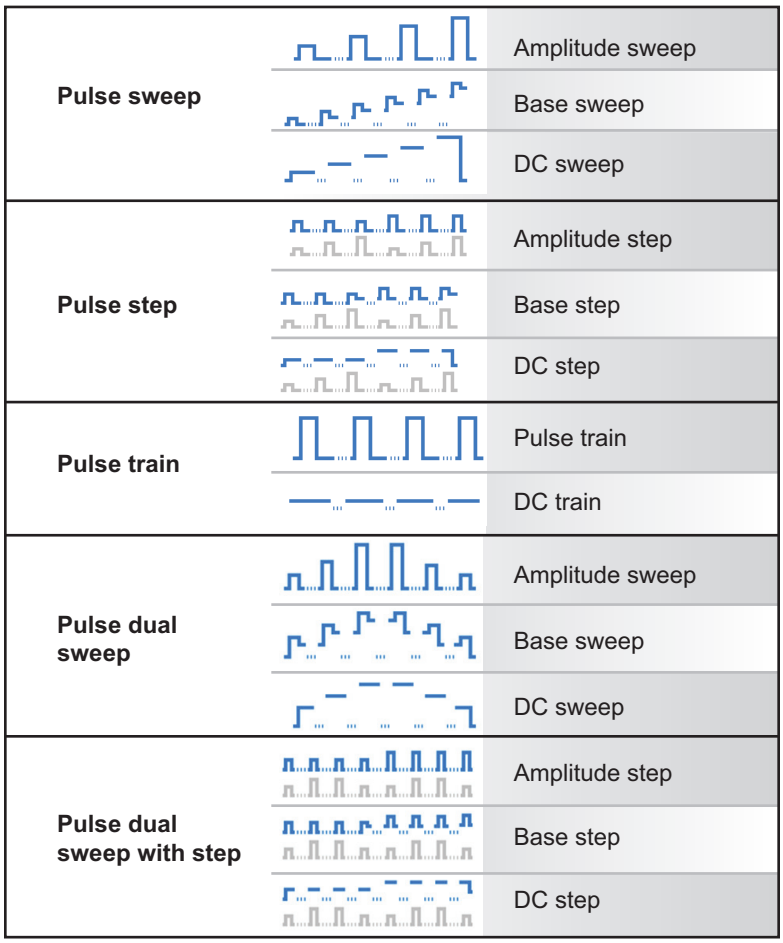
Pressing the Settling Times button displays a window that provides typical minimum timing recommendations for the pulse timing parameters. Pulse timing parameters are set later in this procedure in Step 6) Configure pulse timing. Also see PMU minimum settling times versus current measure range for more information.

NOTE *See Current measurement ranges for the PMU and Current measurement ranges for the PMU with RPM for more information on the current measure ranges.*

Step 4) Configure the selected forcing function

Figure 16-30 shows examples of the available pulse forcing functions.

Figure 16-30
Pulse forcing functions



Depending on the selected forcing function, perform one of the following procedures:

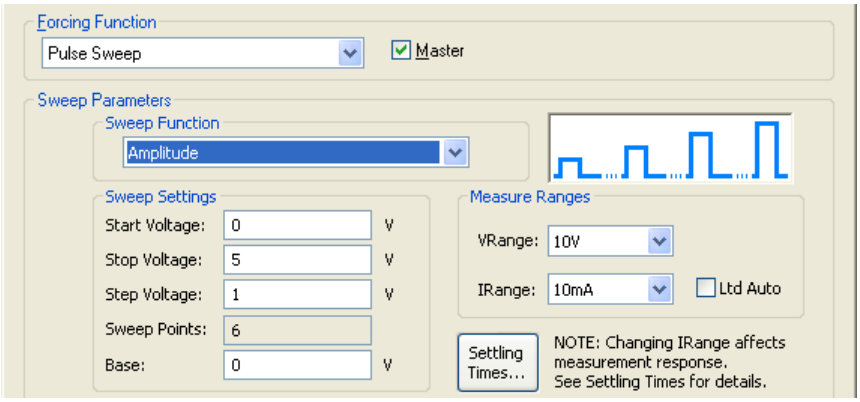
- For pulse sweep, step sweep, or pulse dual sweep, see Pulse sweep and pulse step forcing functions configuration.
- For pulse train, see Pulse train forcing function configuration.

Figure 16-30 illustrates one pulse per period representing that the number of pulses parameter is set to one (1). The ellipsis ("...") between each burst of pulses indicates additional time that the pulse channel is outputting 0 V DC (the pulse channel is not pulsing). This time allows for analog-to-digital (A/D) sample processing and, if enabled, measure ranging and LLEC. For more information on LLEC, see Load line effect compensation (LLEC) for the PMU on page 16-60 and Figure 16-65.

Pulse sweep and pulse step forcing functions configuration

If the pulse sweep or pulse step forcing function is selected, select the sweep or step function (**Amplitude**, **Base**, or **DC)**. Figure 16-31 shows an example of an amplitude sweep.

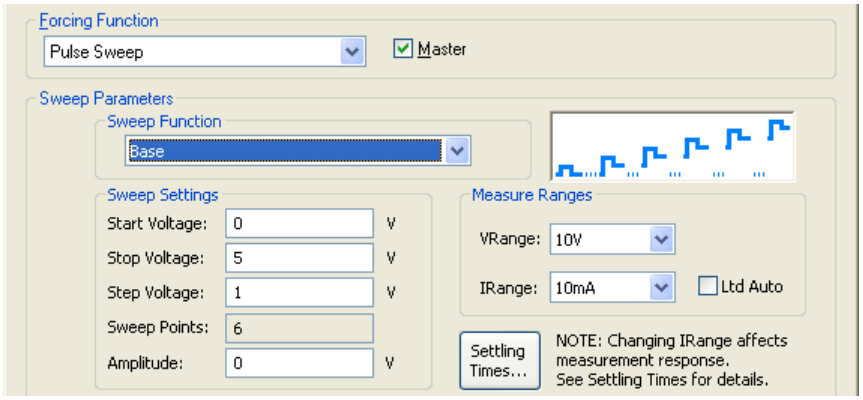
Figure 16-31
Amplitude sweep



For an amplitude sweep, set the start, stop, and step voltages for the amplitude. Also set the base voltage, which remains fixed during the sweep. The base value is the voltage offset (from 0 V) that is the reference for the pulse amplitude and is output during the entire pulse period.

Figure 16-32 shows an example of a base sweep.

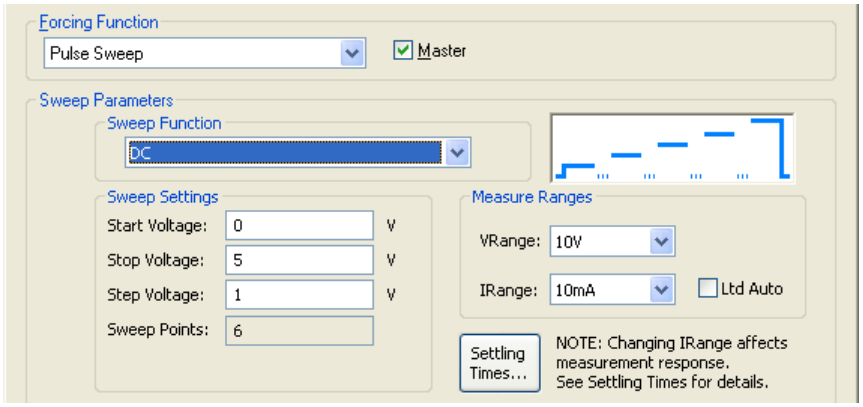
Figure 16-32
Base sweep



For a base sweep, set the start, stop, and step voltages for the base. Also set the amplitude voltage, which remains fixed during the sweep.

Figure 16-33 shows an example of a DC sweep.

Figure 16-33
DC sweep

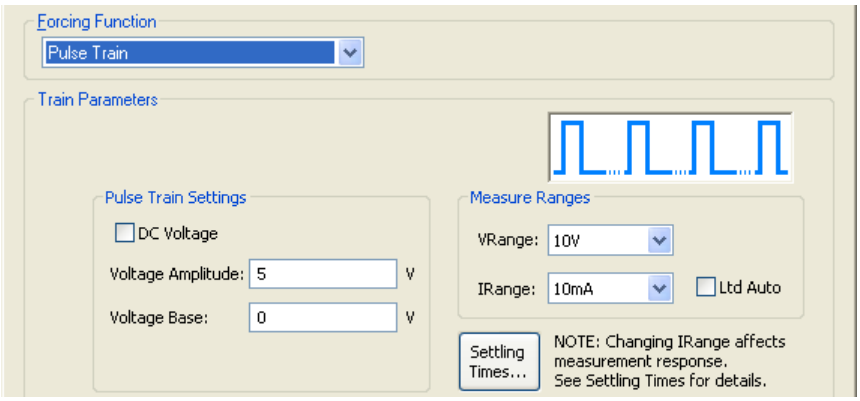


For a DC sweep, set the start, stop, and step voltages.

Pulse train forcing function configuration

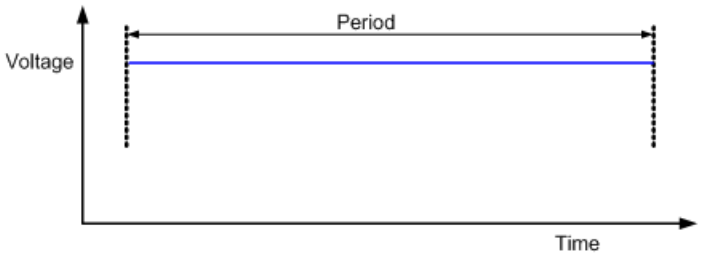
If the pulse train forcing function is selected, set the amplitude and base voltages (see Figure 16-34) for each pulse output by the PMU.

Figure 16-34
Pulse train



Selecting **DC Voltage** disables the amplitude setting. The PMU will output the DC base voltage. Figure 16-35 shows a representation of a DC voltage waveform.

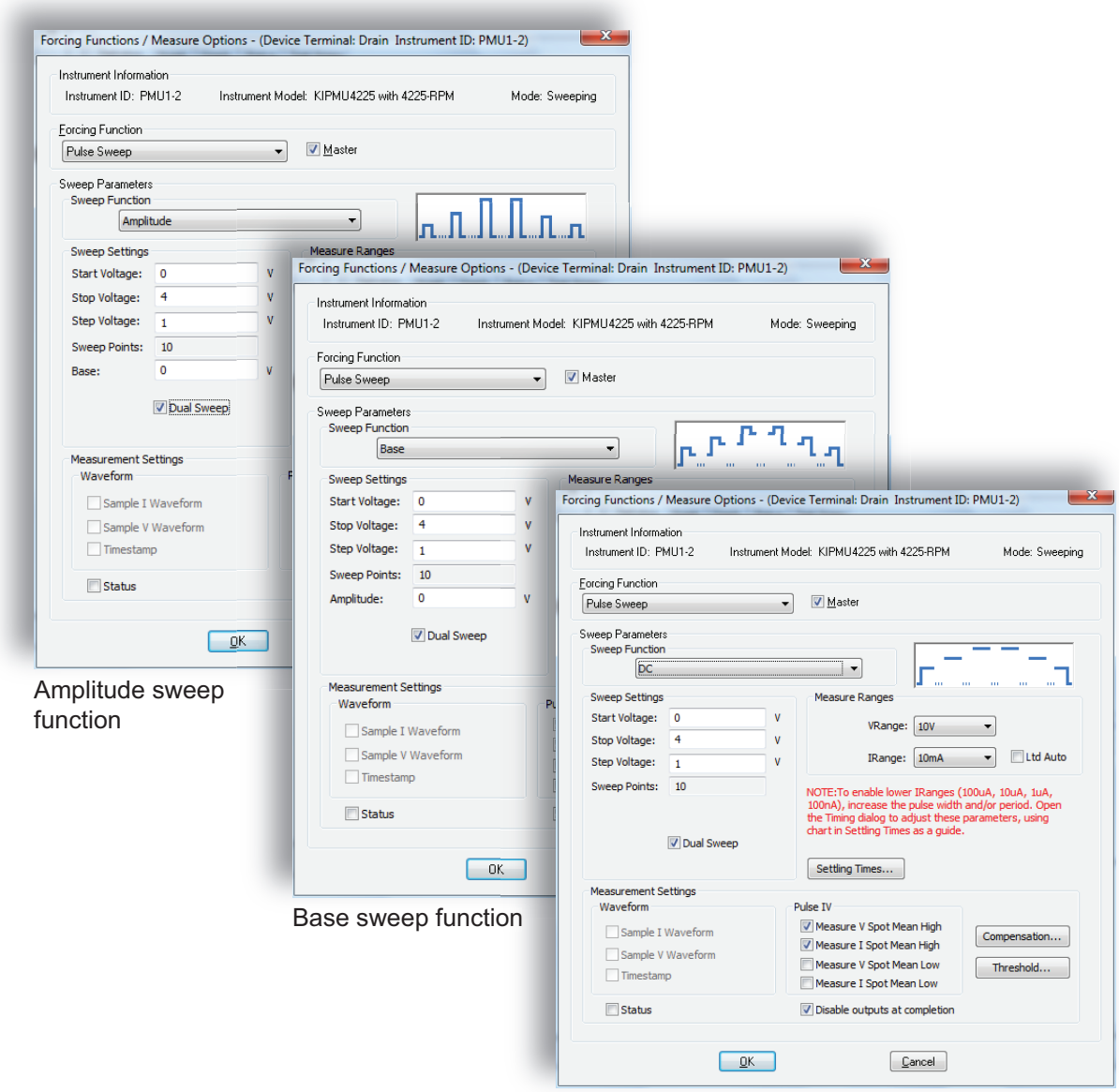
Figure 16-35
DC voltage waveform



Dual Sweep Option

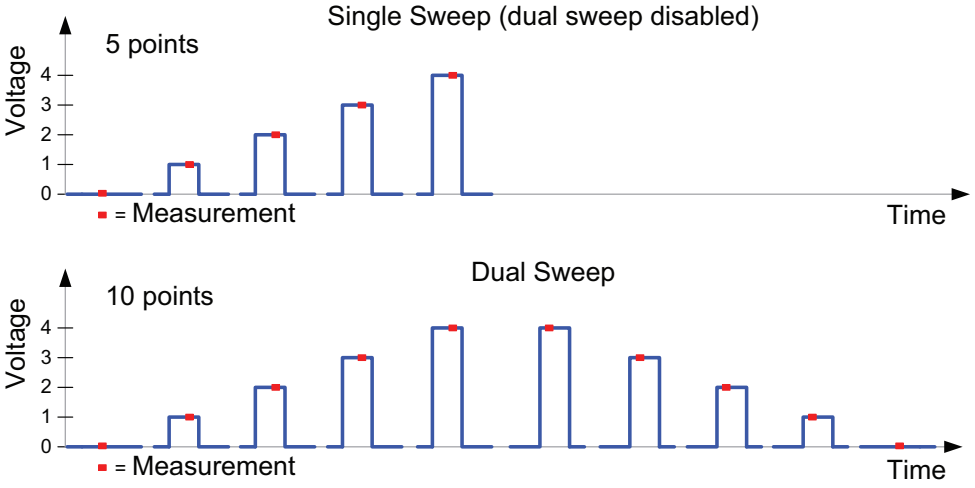
The amplitude, base and DC sweeps have an option for Dual Sweep (Figure 16-37). Figure 16-37 shows the difference between a single pulse sweep and a dual pulse sweep. Figure 16-32 shows a summary of all PMU-based forcing functions.

Figure 16-36
Figure 16-37B. Dual Sweep option for Amplitude, Base and DC sweeps



NOTE *This following figure illustrates a linear voltage sweep; 0 V to 4 V in 1 V steps.*

Figure 16-37
Figure 16-37C. Single and dual pulse amplitude sweep examples



Step 5) Configure measurements

Depending on the selected test mode (as set in Step 1) Select the test and measure modes), perform one of the following procedures:

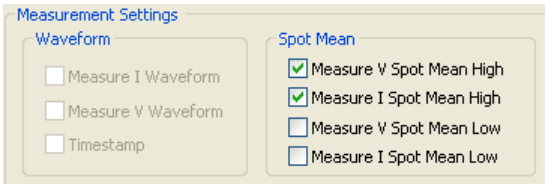
- For pulse IV test mode, see Spot mean measurement configuration.
- For waveform capture test mode, see Waveform capture measurement configuration.

There are also miscellaneous measurement settings that are common to both test modes (see Miscellaneous measurement settings).

Spot mean measurement configuration

If the pulse IV test mode is selected, the settings for spot mean will be enabled. In the Forcing Functions / Measure Options window (), select the spot mean measurements to be performed.

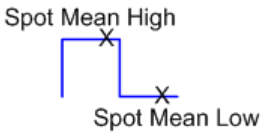
Measurement settings for spot mean



- **Measure V Spot Mean High**: When selected, perform voltage measurements on the amplitude.
- **Measure I Spot Mean High**: When selected, perform current measurements on the amplitude.
- **Measure V Spot Mean Low**: When selected, perform voltage measurements on the base level.
- **Measure I Spot Mean Low**: When selected, perform current measurements on the base level.

Figure 16-38 shows an example of a spot mean measurement on pulse high (amplitude) and pulse low (base level). Figure 16-4 shows how spot mean measurements are performed.

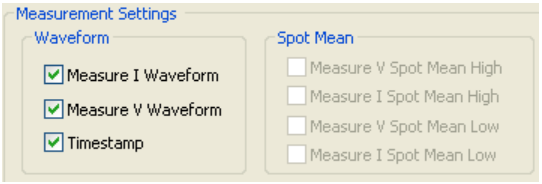
Figure 16-38
Spot mean measurements



Waveform capture measurement configuration

If the waveform capture test mode is selected, the settings for Waveform will be enabled. In the Forcing Functions / Measure Options window (see Figure 16-39), select the waveform capture measurements to be performed and enable or disable time stamps. Figure 16-5 shows how waveform capture measurements are performed.

Figure 16-39
Measurement settings for waveform capture

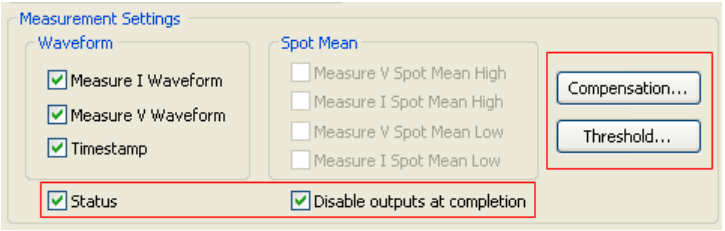


- **Measure I Waveform**: When selected, perform current measurements on the waveform.
- **Measure V Waveform**: When selected, perform voltage measurements on the waveform.
- **Timestamp**: When selected, each measurement will have a time stamp. With Timestamp enabled, the waveform will be graphed (voltage/current versus time) in the Graph tab.

Miscellaneous measurement settings

Miscellaneous measurement settings are shown in Figure 16-40.

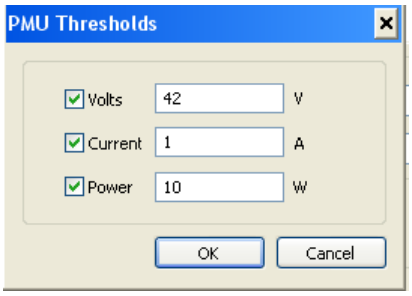
Figure 16-40
Miscellaneous measurement settings



- **Status**: When selected, KITE records measurement status information in the Sheet tab when the ITM executes. Status is returned as a 32-bit word. The status code bit map is shown in Table 8-21 in Section 8.
- **Disable outputs at completion**: When selected, the PMU outputs are disabled when the test is completed.
- **Compensation**: Use to enable or disable short connection compensation (see Enabling connection compensation) and load line effect compensation (see Controlling LLEC from an ITM).
- **Threshold**: Voltage, current, and power thresholds can be set for the PMU (see Figure 16-41). If a selected threshold is reached or exceeded, the present sweep is stopped and

testing continues with any subsequent sweeps. This threshold is not a hardware limit or compliance, but actually a comparison of the measurement after the pulse has been applied to the DUT (the threshold value will be exceeded, at least slightly). The degree that the signal exceeds the threshold before the threshold is triggered is a function of the DUT response and test configuration. Generally, smaller voltage step sizes will reduce the amount that the signal exceeds the threshold. It is possible to have more than one threshold specified for each channel.

Figure 16-41
PMU thresholds



NOTE *After completing Step 5, click* **OK** *in the Forcing Functions / Measure Options window to close the window.*

Step 6) Configure pulse timing

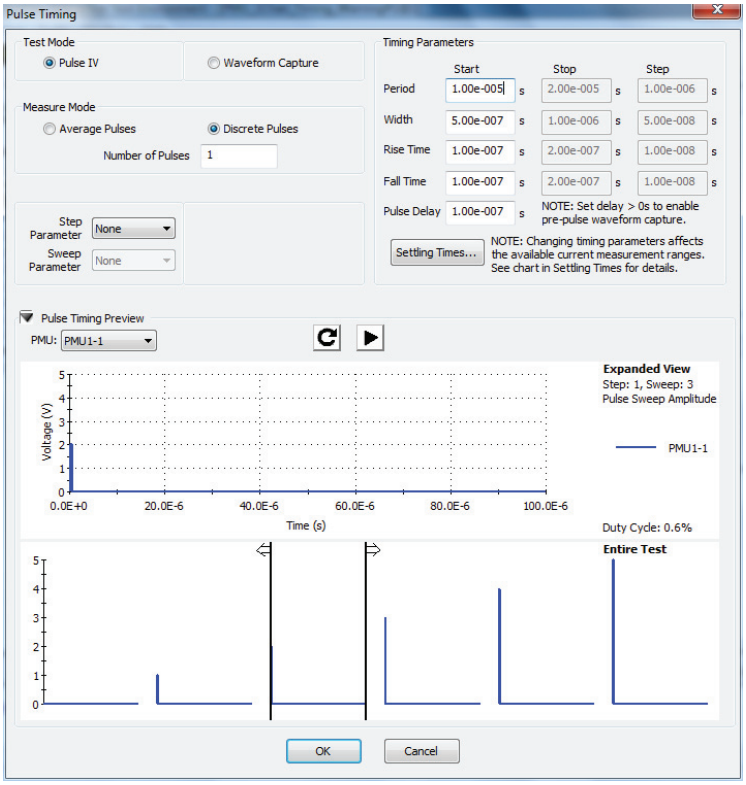
The Pulse Timing Dialog window is used to set pulse period, pulse width, pulse rise and fall times, and pulse delay. The Pulse parameter definitions for these pulse timing settings are provided in Section 11.

Depending on the presently selected test mode and forcing function, you have the option to select an active timing parameter for stepping and sweeping.

Perform the following steps to set pulse timing:

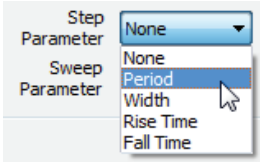
1. In ITM Definition tab (see Figure 16-22), click **Timing** to open the Pulse Timing Dialog window (shown in Figure 16-42).

Figure 16-42
Pulse Timing Dialog window



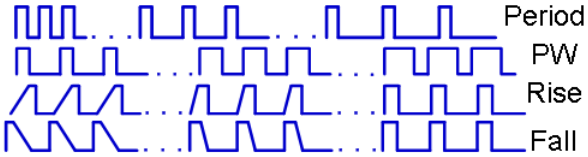
2. Use an active drop-down menu to select a step/sweep parameter, or select **None**. Figure 16-43 shows the active drop-down menu for a sweep parameter. As shown, you can select pulse period, pulse width (PW), pulse rise time, or pulse fall time.

Figure 16-43
Selecting a step/sweep parameter



The timing parameters can only be stepped if the forcing function is either set to pulse sweep or pulse train (not pulse step). The timing parameters can only be swept if the forcing function is either set to pulse step or pulse train (not pulse sweep). Figure 16-44 shows parameter stepping/sweeping examples.

Figure 16-44
Parameter stepping/sweeping examples



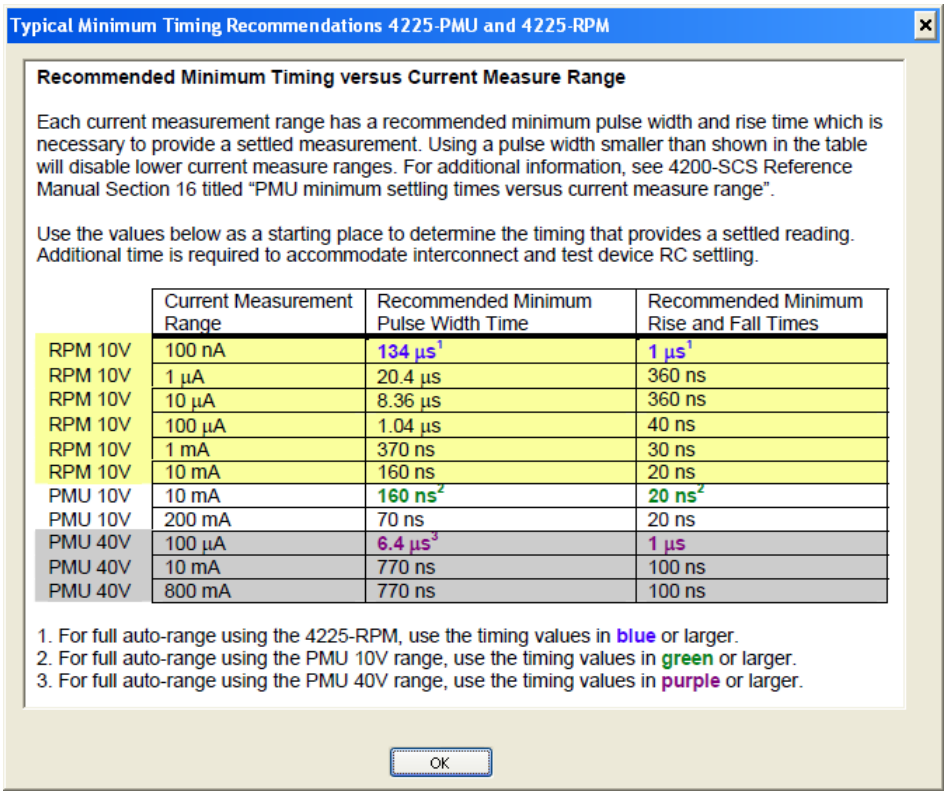
- Period: The pulse period steps or sweeps from a short period to a long period (or from a long period to a short period).
- PW: The pulse width (PW) steps/sweeps from a short pulse width to a long pulse width (or from a long width to a short width).

- Rise: The pulse rise time steps/sweeps from a long rise time to a short rise time (or from a long rise time to a short rise time).
- Fall: The pulse fall time steps/sweeps from a long fall time to a short fall time (or from a long fall time to a short fall time).

Figure 16-44 illustrates three pulses per period representing that the number of pulses parameter is set to three (3). This is different from Figure 16-30 that illustrated one pulse per period (the number of pulses parameter for Figure 16-30 being set to one). The ellipsis ("...") between each burst of pulses indicates additional time that the pulse channel is outputting 0 V DC (the pulse channel is not pulsing). This time allows for analog-to-digital (A/D) sample processing and, if enabled, measure ranging and LLEC. For more information on LLEC, see Load line effect compensation (LLEC) for the PMU on page 16-60 and Figure 16-65.

3. Click the **Settling Times** button (located near the bottom of the Pulse Timing Dialog window) to view the Typical Minimum Timing Recommendations window (see Figure 16-45). These are the recommended minimums for each current measure range. Longer times may be required to accommodate DUT and/or interconnect settling.

Figure 16-45
Typical Minimum Timing Recommendations window



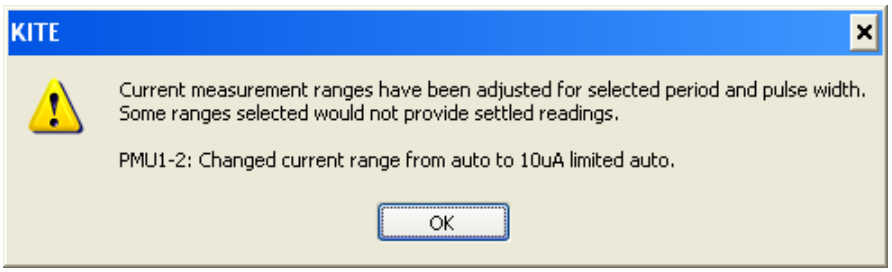
4. Using the timing recommendations shown in Figure 16-65 as a guide, set the **Timing Parameters** that are appropriate for the test. To minimize self-heating effects, set a pulse period that is 10 to 100 times longer than the pulse width to produce a duty cycle that is 1 percent to10 percent.

   When using autorange with an RPM connected, use the timing values on the top row (or longer/slower values). If using limited autorange, start with the timing values for that range.

NOTE *If pulse width or period timing parameters are too narrow for a chosen measure range, a message will be displayed showing the change to the current measure*

range for each effected channel (Figure 16-46). To avoid these changes, follow the guidelines shown in Figure 16-45. Step 3) Set the measure ranges of this procedure sets the current measure range. See PMU minimum settling times versus current measure range for more information.

Figure 16-46
PMU Timing Dialog message for current measure range availability

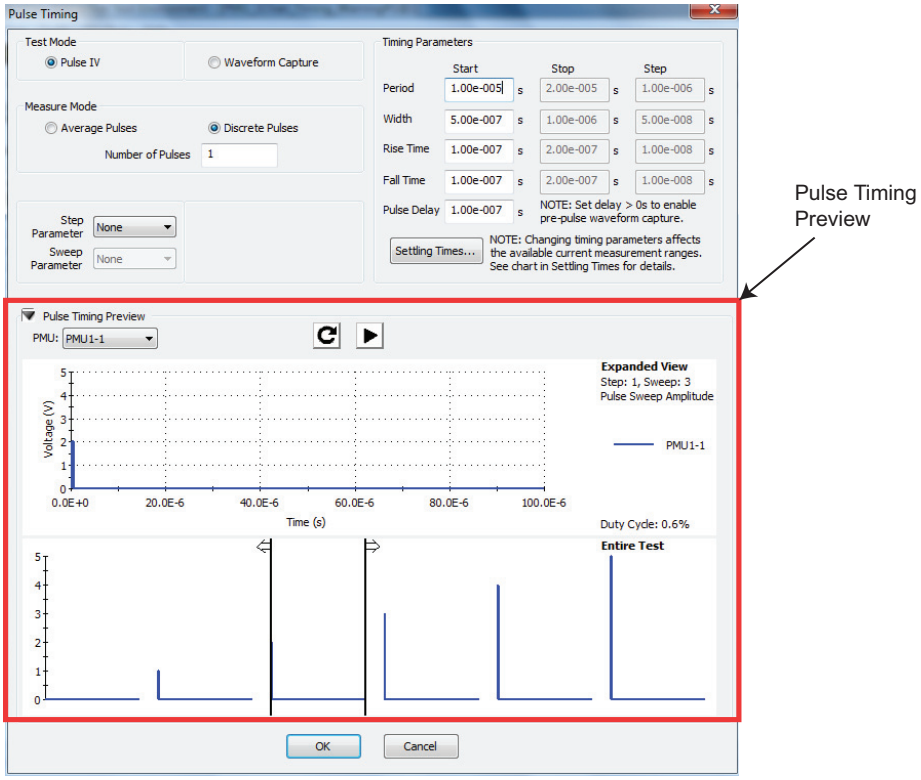


5. Click **OK** to close the Pulse Timing Dialog window.

PMU pulse timing preview

Before executing a test, you can preview a waveform generated using the settings. This can help you understand the behaviors of the various test settings without applying signals to the test device. The Pulse Timing Preview is located in the lower part of the Pulse Timing dialog box (see Figure 16-47).

Figure 16-47
Pulse Timing dialog, preview area

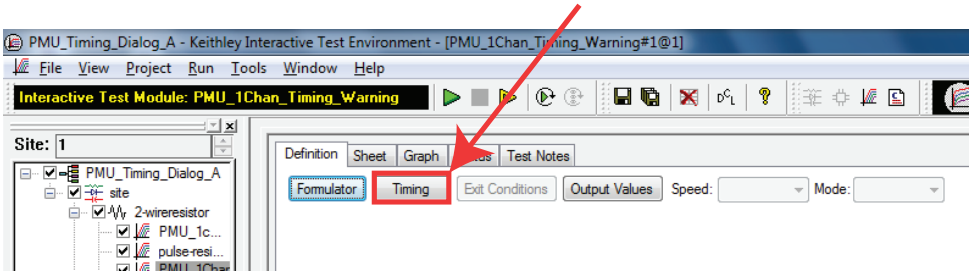


The preview displays representations of the waveforms for the different PMU channels in the test. The Forcing Function / Measure options dialog (see Figure 16-49) defines the channel's voltage levels and forcing functions. The illustrated pulse waveforms are defined by the Pulse Timing dialog settings for the specific PMU. This graphical depiction provides a representation of parameter values (no signals are output to the test device). It is not intended to be a strict definition of the actual number of pulses or pulse voltages that will be applied to the device under test (DUT). When the actual test is running, the actual number of pulses or pulse voltages may be different from the preview.

Enabling current measure ranging or load line effect compensation (LLEC) requires additional pulses to what is shown in the preview. To learn about how the PMU handles device under test (DUT) load variation and measure ranging during a test, see "How LLEC adjusts pulse output to the target levels" on page 16-61.

To open the Pulse Timing dialog (Figure 16-50), click the **Timing** button on the on the PMU Definition (Figure 16-48).

Figure 16-48
Timing button (PDU Definition tab)



Review the following examples to understand the various features of the Pulse Timing Preview feature.

PMU amplitude sweep example (one-channel)

This example sweep has the voltages and forcing functions shown in Figure 16-49. This figure illustrates the configuration for a pulse amplitude sweep using a single PMU channel.

Figure 16-49
One-channel PMU amplitude sweep

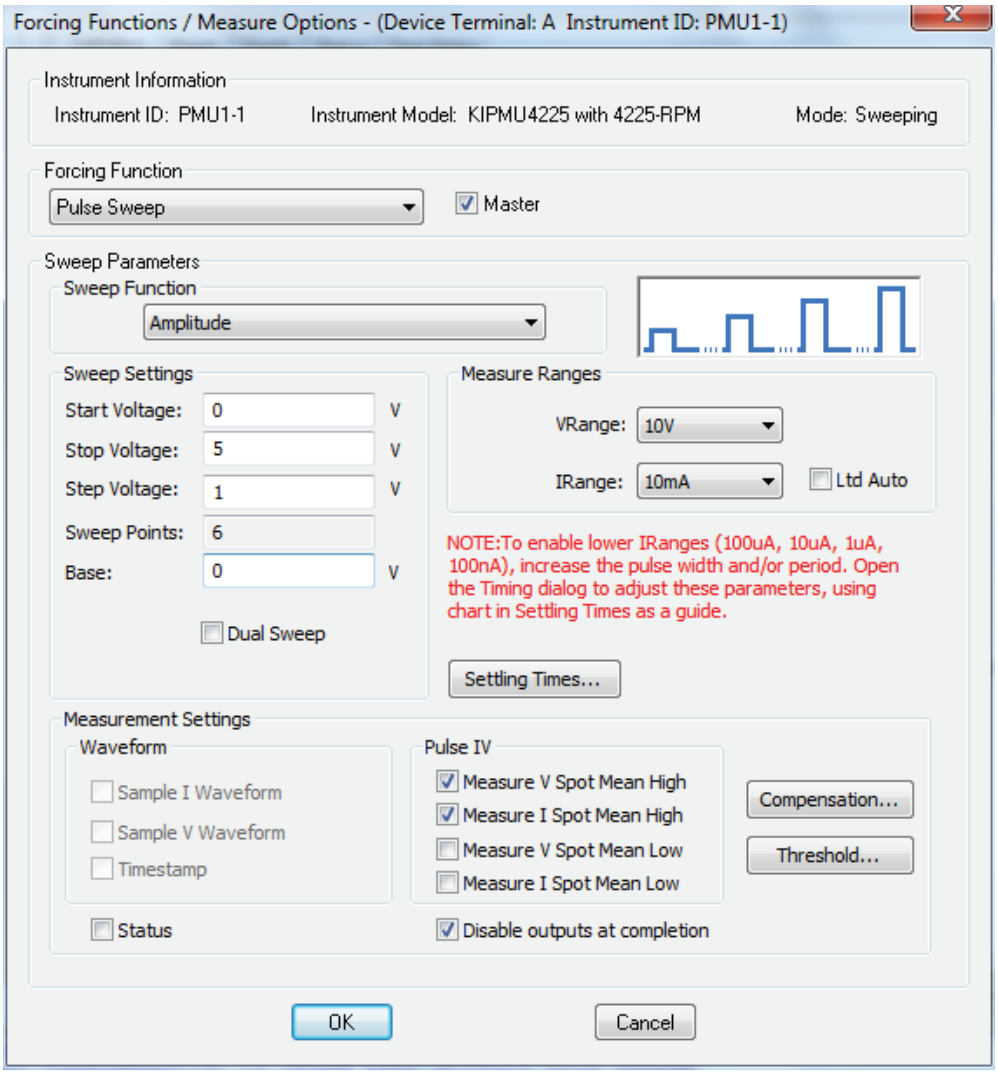
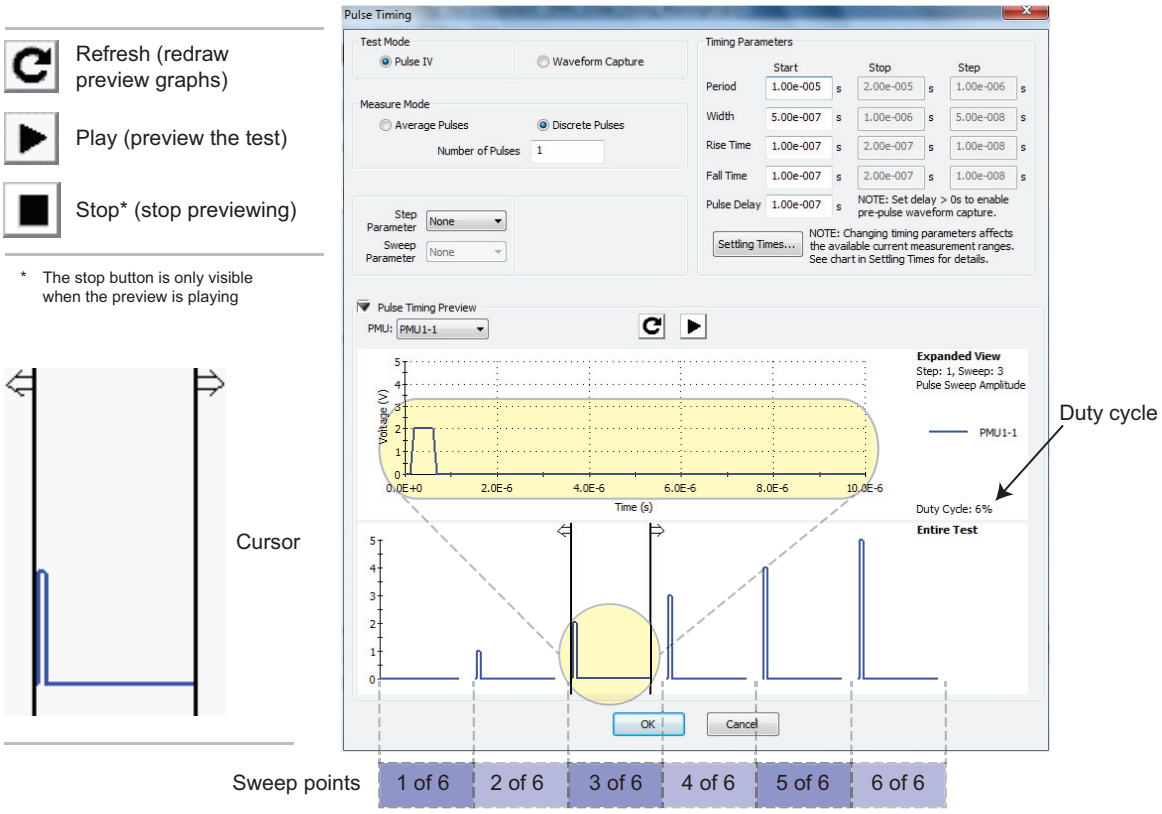


Figure 16-50 illustrates the configuration of a six-step pulse amplitude sweep. In this sweep, the Pulse Timing dialog and the Forcing Function / Measure Options dialog (see Figure 16-49) define the test parameters.

In the Pulse Timing Preview, there are two graphs. The bottom graph, labeled "Entire Test," shows the complete test. This graph shows each sweep and step point for the entire test. The graphed points also reflect the settings from the Force Functions / Measure Options dialog for each PMU channel in the test. The cursor (the pair of vertical black lines on the Entire Test graph) defines the upper graph's content (the upper graph is labeled "Expanded View"). Note that the test from Figure 16-49 defines the six sweep points that are shown in the bottom graph of Figure 16-50 as six pulse periods. In this example, the pulse periods have voltage amplitude increasing from 0 V to 5 V (this is defined in Figure 16-49).

The other graph, labeled "Expanded View," provides in depth information on the section designated by the cursor lines (which are located on the graph labeled "Entire Test"). The graphed points reflect the timing parameters (from the upper portion of the Pulse Timing dialog) and settings from the Force Functions / Measure Options dialog for each PMU channel in the test, but only for the area specified by the cursor.

Figure 16-50
Six-point pulse amplitude sweep



Pressing the play button animates the waveform ("play" previews the test settings). The cursor moves from left to right on the bottom graph while displaying each sweep's point waveform within the cursor on the top graph.

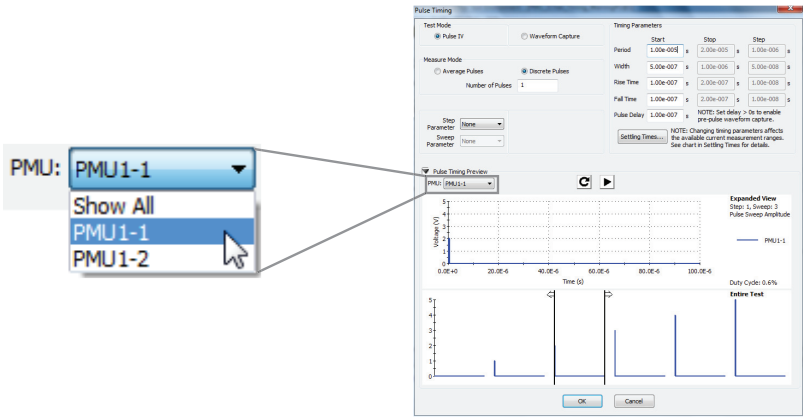
NOTE *Playing the test preview does not output any pulses; it just illustrates how the test will progress as a result of the test parameter settings.*

PMU amplitude sweep and step example (two-channel)

In this two-channel example, one channel is sweeping while the other channel is stepping. The sweeping channel is performing pulse amplitude sweeps with nine sweep points. On each inner loop, PMU1-2 sweeps 0 V through 4 V in 0.5 V increments. The stepping channel is performing pulse amplitude steps with four step points. On the outer loop, PMU1-1 pulses the four steps of 1 V through 2.5 V in 0.5 V increments.

A preview showing all channels is shown in Figure 16-52. You can set the preview to show a single channel or to show all channels in the test. To set the preview, click the PMU Timing preview channel dropdown and select a display (Figure 16-51).

Figure 16-51
PMU Timing preview channel dropdown



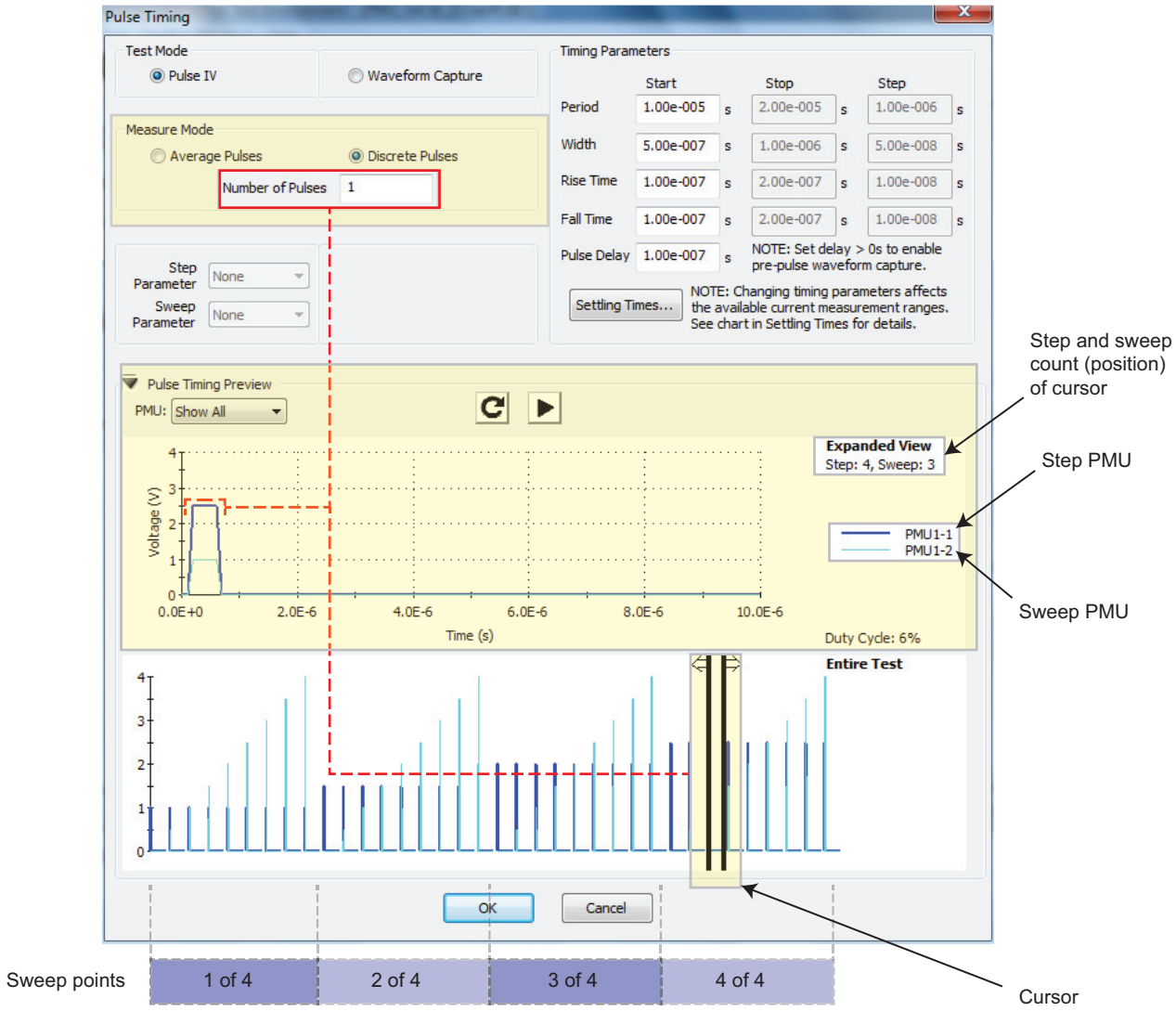
See Table 16-10 for the specific values used in this example. The graph of the entire test (the lower graph of Figure 16-52) shows four steps of nine points each.

Table 16-10
Forcing Functions and Voltages for Figure 16-52

Forcing Function	PMU1-1 (blue waveform)	PMU1-2 (light blue waveform)
	Pulse Amplitude Step	**Pulse Amplitude Sweep**
Start Voltage	1	0
Stop Voltage	2.5	4
Step Voltage	0.5	0.5
Number of steps or sweep points	4	9

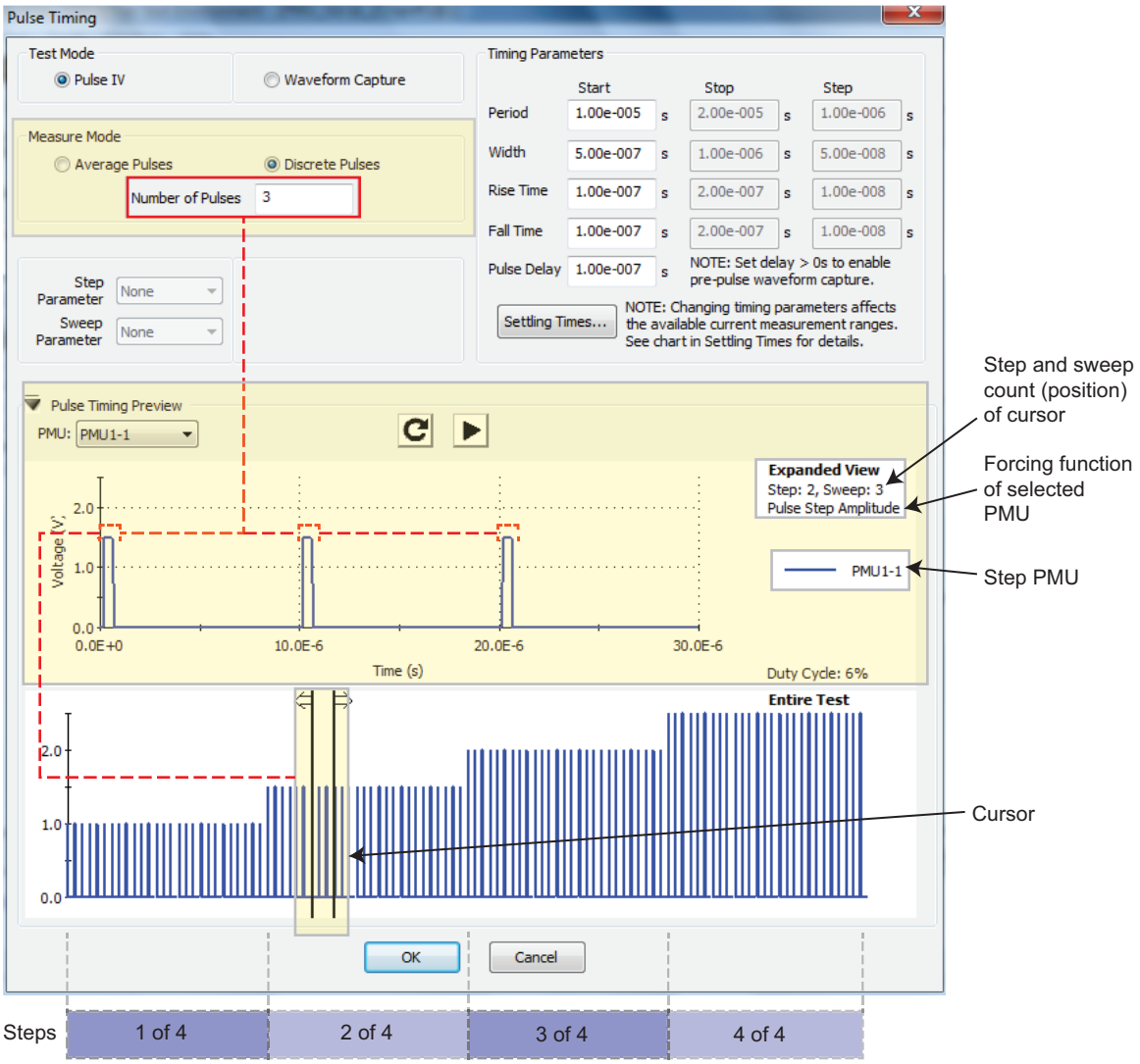
The sweep and step count displays the cursor's position within the overall test. This example has the number of pulses set to one. Set this by changing the **Number of Pulses** parameter in the Measure Mode area of the Pulse Timing dialog.

Figure 16-52
Two-channel test (PMU1-1 stepping and PMU1-2 sweeping)



Changing the number of pulses to three (instead of one) and only displaying one channel (PMU1-1, instead of Show All), causes changes to the preview of the waveform (see Figure 16-53). Each point in the step now uses three periods, so there are three pulses shown in the Expanded View. In the Expanded View, the graph's x-axis shows the time in seconds. This is three times longer than the Expanded View graph when number of pulses is set to one (see Figure 16-52).

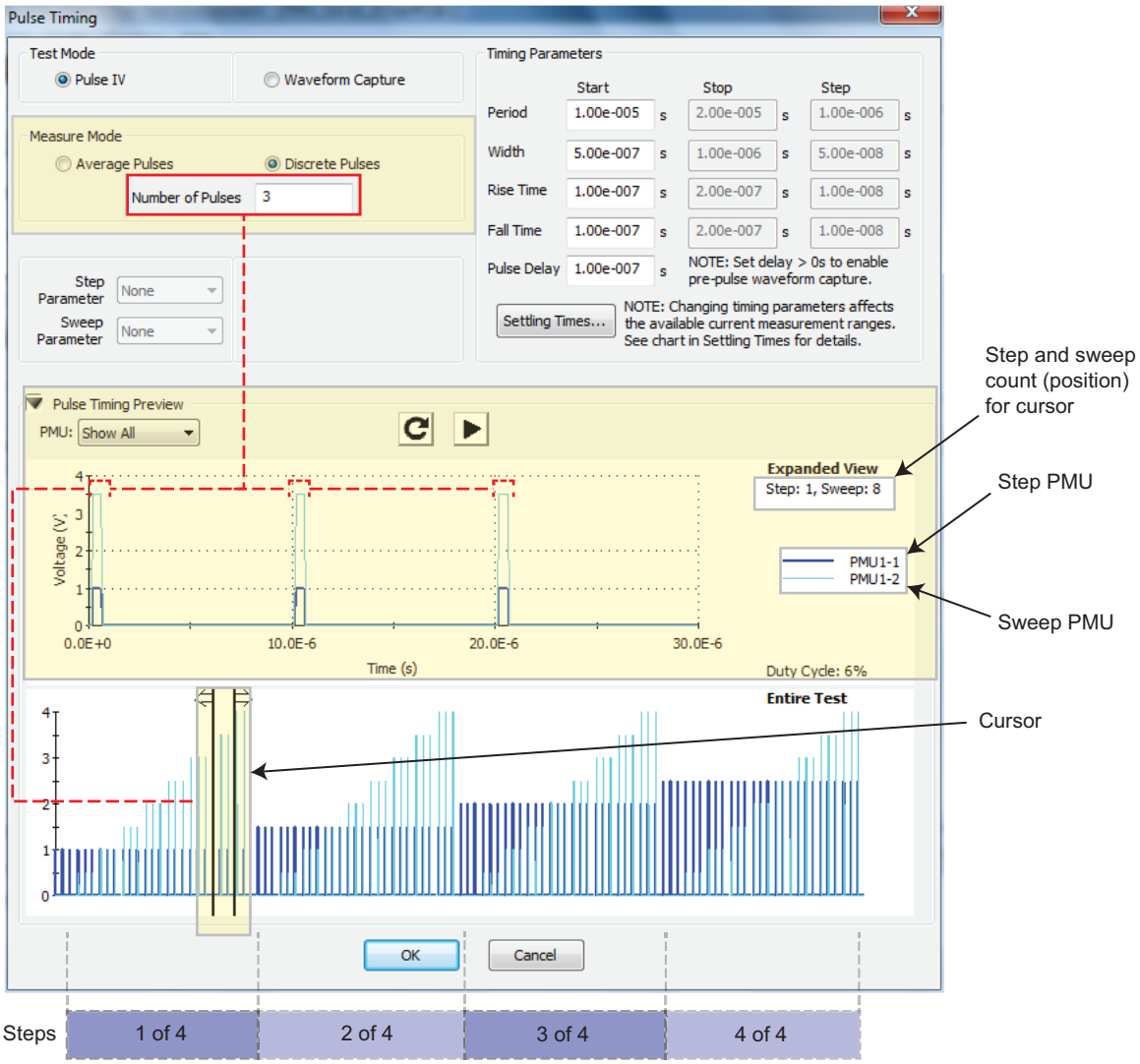
Figure 16-53
Two-channel test (stepping and sweeping) with three points (only PMU1-1 displayed)



Configuring the preview to show all channels in the test causes displays both the sweeper and stepper for the three-pulse test (see Figure 16-54).

The Number of Pulses parameter (see Figure 16-50) determines the number of pulses that will be output and measured for each attempted point in the sweep or step. Figure 16-52 shows PMU1-1 (the stepper from Figure 16-52) with the number of pulses changed to three. Notice that each sweep point of the displayed waveform in Figure 16-52 has three pulses (number of pulses set to three); the light blue waveform in Figure 16-52 has one pulse (number of pulses set to one). Because the cursor in the lower graph always contains the complete waveform of each point in a test, the top graph shows the three pulses. Figure 16-54 shows both channels for the sweep and step test with the number of pulses set to three. Note that the value for the number of pulses does not affect the number of sweep or step points in the test.

Figure 16-54
Two-channel test with three points (PMU1-1 and PMU1-2 displayed)



Higher channel count test example

This example shows how the Pulse Timing Preview is useful for higher channel count tests. Figure 16-55 shows a four-channel test, reflecting the Forcing Function and voltages given in Table 16-11. When the channel count is higher, using the zoom feature provides a more detailed view of the individual channel.

Expanded View zoom

Press and hold the left mouse button to drag the pointer and define the area to magnify (the cursor turns into a magnifying glass while dragging). See Figure 16-56 for an example. Note that each channel has a unique color and line width. When the channels overlap, narrower lines are shown on top of the wider lines. To return to normal magnification, double-click on the graph (alternatively, click the Refresh waveform button). Multiple levels of zoom are supported, with a double-click returning each level. Note that the Refresh waveform button restores normal magnification completely (shows the Entire Test waveform).

Table 16-11
Forcing Functions and Voltages for the 4-channel test show in Figure 16-55

	PMU1-1	PMU1-2	PMU2-1	PMU2-2
Forcing function	**Pulse amplitude step**	**Pulse amplitude sweep**	**Pulse voltage train**	**Pulse voltage train**
Start voltage	1.5	0	1.5	0.5
Stop voltage	3	4	*	*
Step voltage	0.5	0.0	*	*
Number of points	4	5	*	*
* Not applicable. The pulse voltage trains are fixed pulse voltage levels; they do not vary during the sweep or step points.				

NOTE *See Table 16-11 for the key test parameters used in the following figure.*

Figure 16-55
Four-channel sweep and step (2 pulse trains)

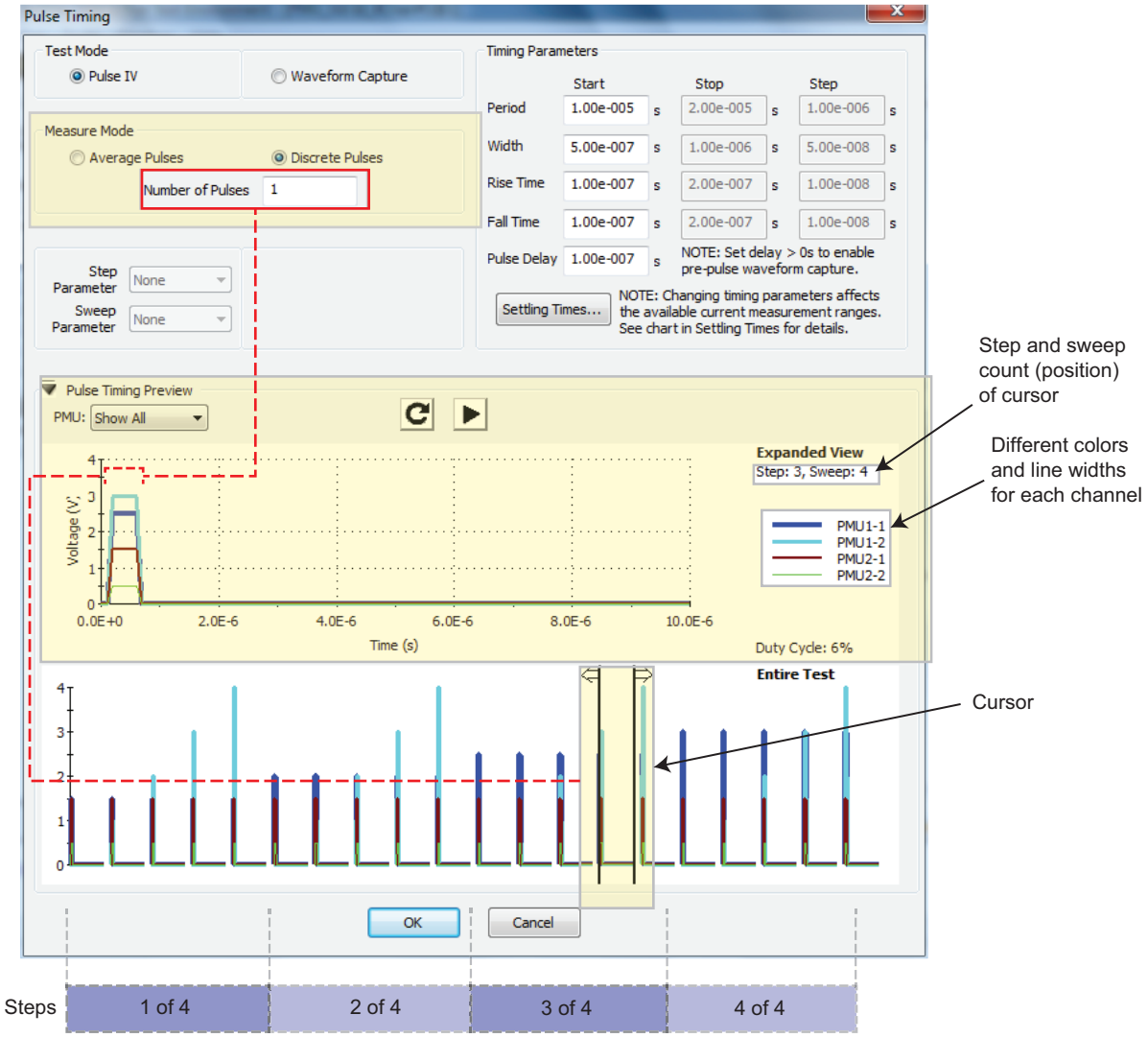
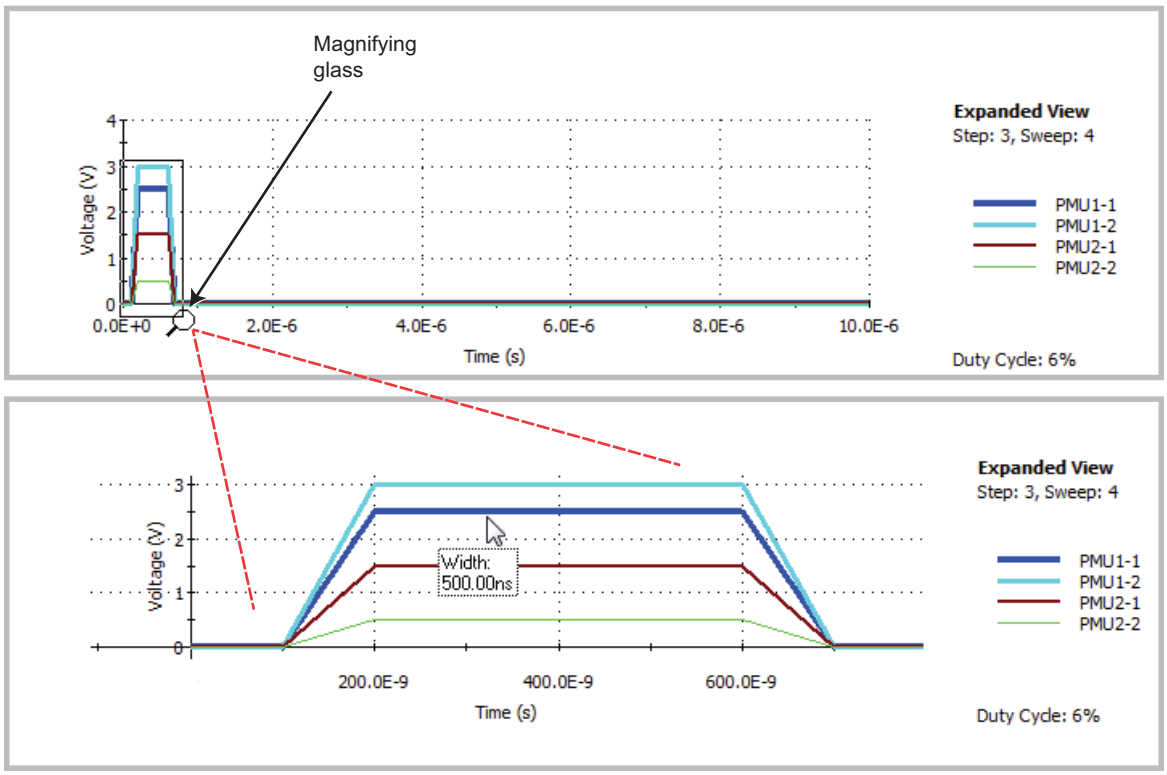


Figure 16-56
Zooming (Expanded View graph)



Scrolling the magnified area

To scroll (or move) the viewable area of the graph after you zoom in on a graph, hold down the shift key on the Model 4200-SCS keyboard while grabbing (left click-and-holding and moving) the graph.

Hovering

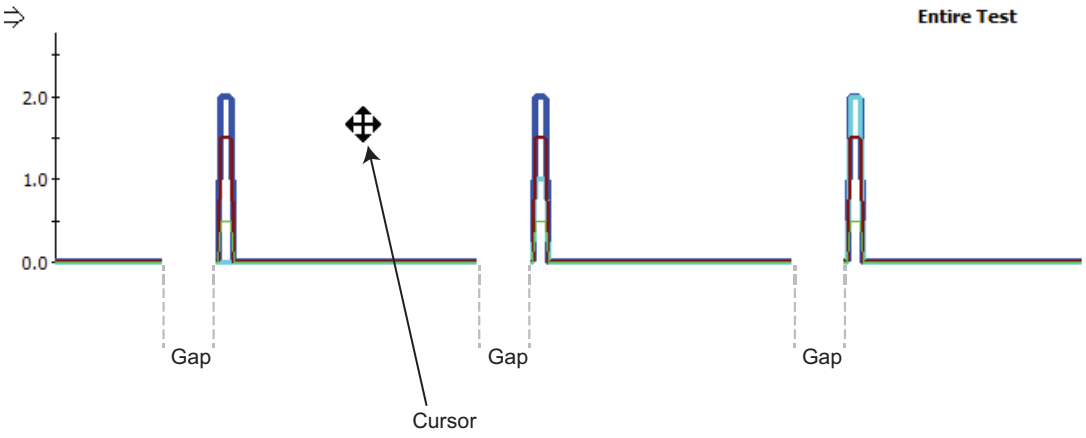
To display the timing information associated with a pulse segment, hover the mouse pointer over that portion of the waveform (see the lower portion of Figure 16-56). The mouse pointer in the lower Expanded View of Figure 16-56 shows a pulse width for PMU1-1 as 500.00 ns. The width timing shown is for Full Width Half Maximum (FWHM) "Pulse width" on page 11-44; rise and fall times are 0-100 percent.

Entire Test zoom

The zoom feature is also supported on the lower Entire Test waveform graph in the same manner as in the Expanded View. After you zoom in, you can also scroll (or move) the viewable area of the graph. See Figure 16-57 for a view of moving the lower graph using the mouse pointer. Note the gaps between the pulse waveforms shown in Figure 16-55; gaps also exist in Figure 16-57. These gaps indicate the time between sweep points where the PMU performs calculations for the test while the pulse channels output 0 V. To learn more about how these gaps relate to how the PMU handles DUT load variation and measure ranging during a test, see "How LLEC adjusts pulse output to the target levels" on page 16-61.

While holding down the **Shift** key on the Model 4200-SCS keyboard and left mouse button, move or scroll a magnified graph by moving the mouse.

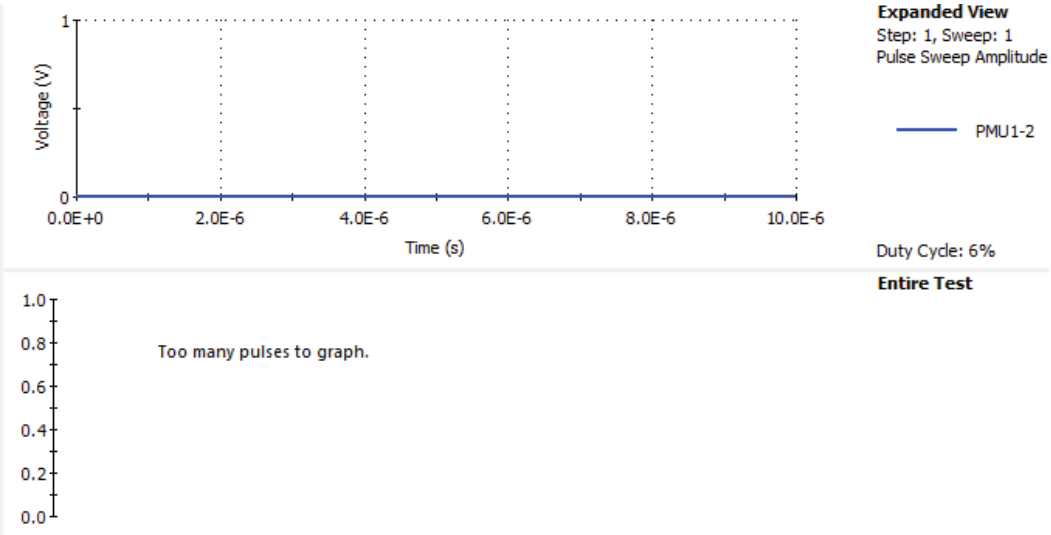
Figure 16-57
Scroll (or move) a magnified entire test graph



Errors

If either graph is not updating as expected, press the refresh button to zoom out and redraw both graphs. It is possible to have a test with too many pulses to be suitably graphed. This may be from too many pulses from a large number of sweep points or step points, or a large value for the number of pulses. Figure 16-58 shows the output "Too many pulses to graph." for the lower graph. Note that this lack of display does not mean that the test will not run. If pressing OK at the PMU Timing dialog does not generate any warning or error messages, then the test is valid and runs (even though the entire test waveform cannot be displayed).

Figure 16-58
Preview error "Too many pulses to graph"



PMU connection compensation

Errors caused by connections and cable length between the Model 4225-PMU and the device under test (DUT) can be corrected by using connection compensation. When connection compensation is enabled, each DUT measurement factors in either the default or measured (custom) compensation values.

You have the option to use either default connection compensation values (PMU or RPM) or custom connection compensation values. The default values can be used for typical connection setups that use the supplied cables. The custom connection compensated values are generated when connection compensation is performed from the KITE interface. The custom values provide optimum compensation. When a test is run, each DUT measurement will factor in the enabled compensation values. If connection compensation is disabled, the compensation values will not be applied to the measurements.

NOTE *For UTM programming, the default connection compensation values can only be selected (enabled) using the pulse_conncomp function, described in Section 8.*

Custom connection compensation values are generated by performing the connection compensation from the KITE interface. Custom connection compensation data is generated for open and short conditions. The compensation values are stored in tables. The custom connection compensation values can be enabled or disabled from a PMU ITM in KITE.

NOTE *For optimum performance, connection compensation should be performed anytime the connection setup is changed or disturbed. Changes in temperature or humidity do not affect connection compensation.*

While Performing connection compensation, status messages (listed in Table 16-12) are generated. Any failures with the compensation process are reported as errors (listed in Table 16-13).

Table 16-12
Custom connection compensation status messages

Open and short status messages:

`Starting PMU Cable Compensation...`
`R = % Ohms`
`PMU Cable Compensation complete.`

`%` = value (V and I measured, Ohms calculated)

Table 16-13
Custom connection compensation error messages

Error messages:

`Maximum Ch x leakage current at 0V exceeded! I = %gA`
`Maximum Ch x pulse leakage current exceeded! I = %gA`
`Ch x voltage offset exceeded at 0V! V = %gV`
`Ch x cable pulse measure current is out of tolerance! I = %gA`
`Ch x cable pulse measure voltage is out of tolerance! V = %gV`
`Ch x pulse voltage exceeded! V = %gV`
`Open cable impedance too low! R = %g Ohms`
`Ohms Cable impedance too high! R = %g Ohms`

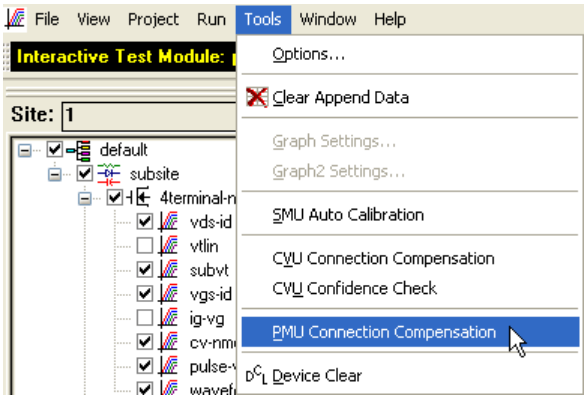
`Ch x` = channel 1 or channel 2 `%g` = double value (V and I measured, Ohms calculated)

Performing connection compensation

Perform connection compensation from the KITE interface as follows:

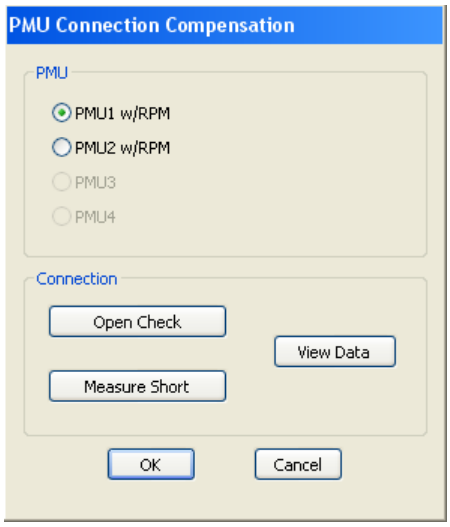
1. At the top of the KITE interface, click **Tools**, and then select **PMU Connection Compensation** (see Figure 16-59).

Figure 16-59
Selecting connection compensation



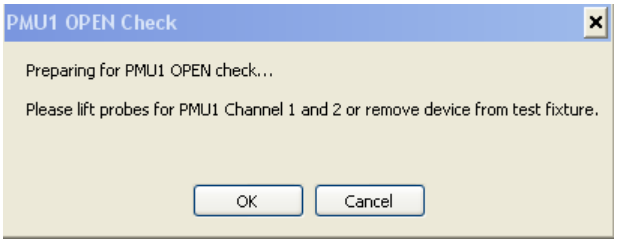
2. In the PMU Connection Compensation window (see Figure 16-60), select a PMU.

Figure 16-60
PMU Connection Compensation window



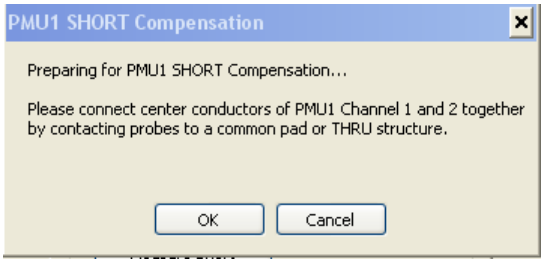
3. In the PMU Connection Compensation window (see Figure 16-60), click **Open Check** to display the PMUx OPEN Check window (see Figure 16-61).

Figure 16-61
PMUx OPEN Check window



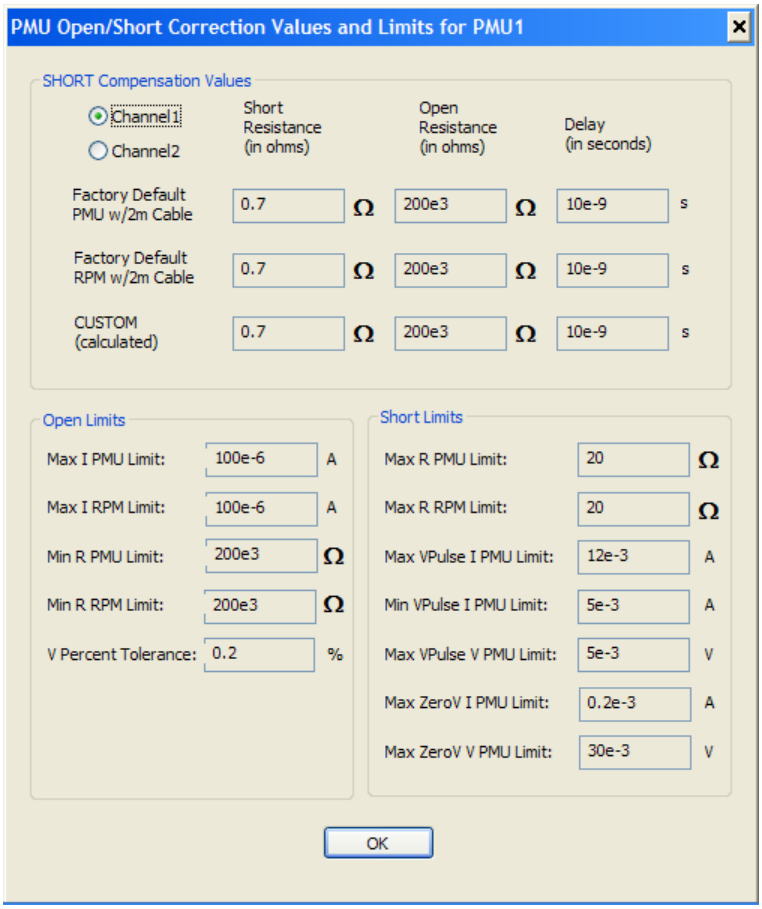
4. Lift the probes off the wafer or remove the DUT from the test fixture.
5. In the PMUx OPEN Check window (see Figure 16-61), click **OK** to perform the open check.
6. In the PMU Connection Compensation window (see Figure 16-60), click **Measure Short** to display the PMUx SHORT Compensation window (see Figure 16-62).

Figure 16-62
PMUx SHORT Compensation window



7. Lower the probes onto a common pad or THRU structure, or replace the DUT in the test fixture with a short.
8. In the PMUx SHORT Compensation window (see Figure 16-62), click **OK** to perform short compensation.
9. To view the compensation data, click **View Data** in the PMU Connection Compensation window (see Figure 16-60). The PMU Open/Short Correction Values and Limits for PMUx is shown in Figure 16-63.

Figure 16-63
PMU Open/Short Correction Values and Limits for PMU window



Short Compensation Values area

At the top of the window in Figure 16-63, select which data to view (Channel 1 data or Channel 2 data). If your test setup used both PMU channels (example shown in Figure 16-10), you will have new custom data for both channels.

This Short Compensation Values area shows compensation values for short, open, and delay. First row data shows default values for the PMU only. Second row data shows default values for the PMU with RPM. Third row data shows the measured (custom) values acquired by performing connection compensation.

Open Limits and Short Limits areas

The Open Limits and Short Limits data and any reported errors (see Table 16-13) is provided to help you ascertain why any measured (custom) open or short value is out of limits.

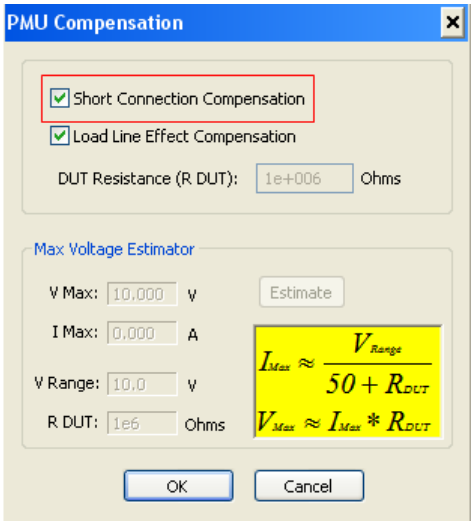
10. When finished viewing the compensation data, click **OK** to close the window.
11. In the PMU Connection Compensation window (see Figure 16-60), click **OK** to close the window.

Enabling connection compensation

Connection compensation data is only applied to DUT measurements when connection compensation is enabled. Connection compensation for custom compensation data is enabled from a PMU ITM in the KITE interface as follows:

1. From the definition tab for an ITM, click the **FORCE / MEASURE** button (see Figure 16-22) for a PMU to open the FORCING FUNCTIONS / MEASURE OPTIONS window (see Figure 16-27).
2. Click the **Compensation** button to open the PMU Compensation window (see Figure 16-64).

Figure 16-64
Enabling connection compensation



3. To enable connection compensation, insert a check mark in the box for **Short Connection Compensation** and click **OK**.

Removing the check mark disables all connection compensation data. When disabled, connection compensation values are not applied to DUT measurements.

NOTE *For UTM programming, default or custom connection compensation data can be enabled (or disabled) using the pulse_conncomp function, described in Section 8.*

Load line effect compensation (LLEC) for the PMU

NOTE *Load line effect compensation (LLEC) is only performed for standard pulse IV testing using PMU measure ranges. It is not performed when using Model 4225-RPM measure ranges (≤1 mA). The active RPM circuitry provides its own analog LLEC (assuming there is a short cable from the RPM to the DUT).*

Load line effect

The basic pulse output system is a series circuit that consists of the pulse generator resistance (fixed at 50 Ω), interconnect (cabling and pin-to-pad) resistance, and the resistance of the DUT. In this series circuit, the sum of the voltage drops across these components is equal to the output voltage of the pulse generator. Therefore, if the resistance of the DUT changes, the voltage seen at the DUT also changes. This effect is called the load line effect. See DUT resistance determines pulse voltage across DUT in Section 11 for details on how the resistance of the DUT affects the voltage across it.

For example, consider a PMU set to output voltage to a 50 Ω load (DUT). For this default setting, the pulse card will output twice the programmed pulse voltage. If the interconnect resistance is negligible (0 Ω), half the pulse card voltage appears across the internal pulse card resistance (50 Ω) and the other half (which is the programmed pulse voltage) appears across the 50 Ω DUT. For example, if the pulse card is programmed to output a 5 V pulse, the pulse card will source a 10 V pulse. Five volts will drop across the internal 50 Ω pulse card resistance and 5 V will appear at the 50 Ω DUT.

Methods to compensate for load line effect

There are three methods to compensate for load line effect:

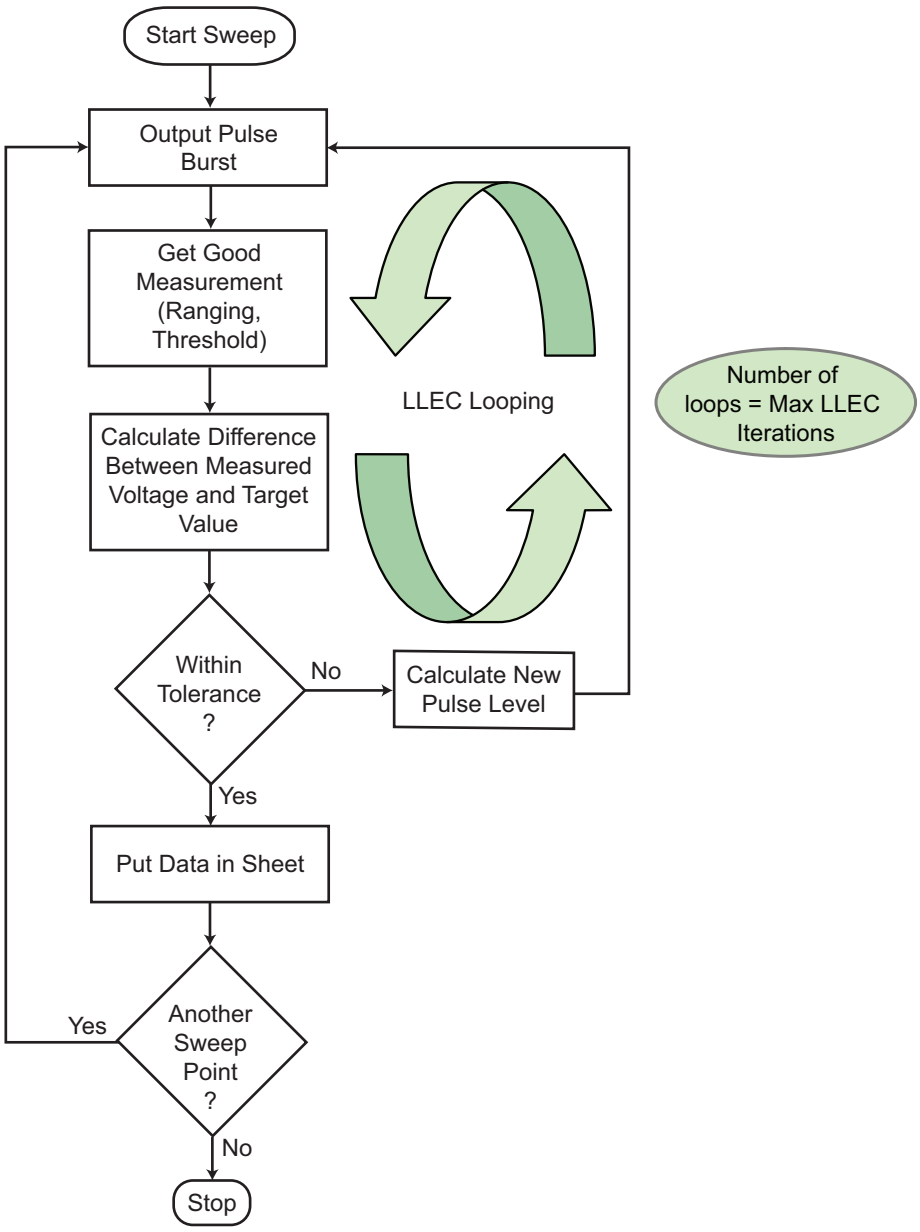
- The PMU has built-in load line effect compensation (LLEC) for the standard, 2-level pulse mode. Ideally (when LLEC is enabled), the PMU adjusts its output levels (within its limits) such that the programmed (target) output voltage appears at the DUT. For ITMs, see Controlling LLEC from an ITM. For UTMs, use the pulse_meas_sm function (in Section 8) to control LLEC.
- Another method to correct for load line effect is to manually adjust the PMU output until the target pulse level is measured across the DUT. For a pulse sweep, this manual process must be repeated for every step in the sweep.
- When not using LLEC, you can manually set the output impedance to match the impedance of the DUT. For example if the impedance of the DUT is 1 kΩ, set the output impedance to 1 kΩ. Maximum power transfer is achieved when the DUT impedance matches the output impedance setting. In a real test environment, you may not know the exact resistance value of the DUT, and you will have the added affect of cabling and pin-to-pad resistance (typically 3 Ω to 10 Ω). The output impedance can be set from an ITM (see Controlling LLEC from an ITM) or from a UTM (see pulse_load function in Section 8).

How LLEC adjusts pulse output to the target levels

LLEC is an algorithm that adjusts the output of a PMU channel. When enabled, the algorithm performs a set number of iterations in an attempt to output the target voltage to the DUT.

The algorithm used for LLEC is shown in Figure 16-65. The diagram shows that the PMU standard pulse source (with measure) uses a burst-measure-analyze-reburst method. This method allows for range changing, threshold comparison, load line effect compensation, and pulse timing. This means there is separation between each set, or burst, of pulses. The number of pulses output for each attempt is controlled by the Number of Pulses setting (see Figure 16-23) for an ITM, or the pulse_meas_timing function for UTMs. Note that LLEC is available only in the standard, 2-level pulse mode. LLEC is not available in the Segment ARB mode.

Figure 16-65
LLEC algorithm for two-level pulsing (PMU)



Note that after the first action, "Output Pulse Burst," all pulse channels in the test stop pulsing and output 0 V while performing the actions in the remaining boxes in the diagram. The time between pulses is determined by the time required to process the measurements and perform the calculations and comparisons shown in Figure 16-65. Wider pulses, longer pulse periods, and a higher number of pulses increases the time between pulses where the output is at 0 V. Note that both Pulse IV and Waveform Capture Test modes use this algorithm and both will output 0 V between pulses for each step in a sweep. For strict control over the pulse voltage versus time, see the Segment ARB feature of the PMU.

The “Get Good Measurement” step shown in Figure 16-65 also must ensure that the current measure range is correct (if ranging is enabled) and check the measured voltage and current against the thresholds. See Step 3) Set the measure ranges for ranging configuration and Miscellaneous measurement settings.

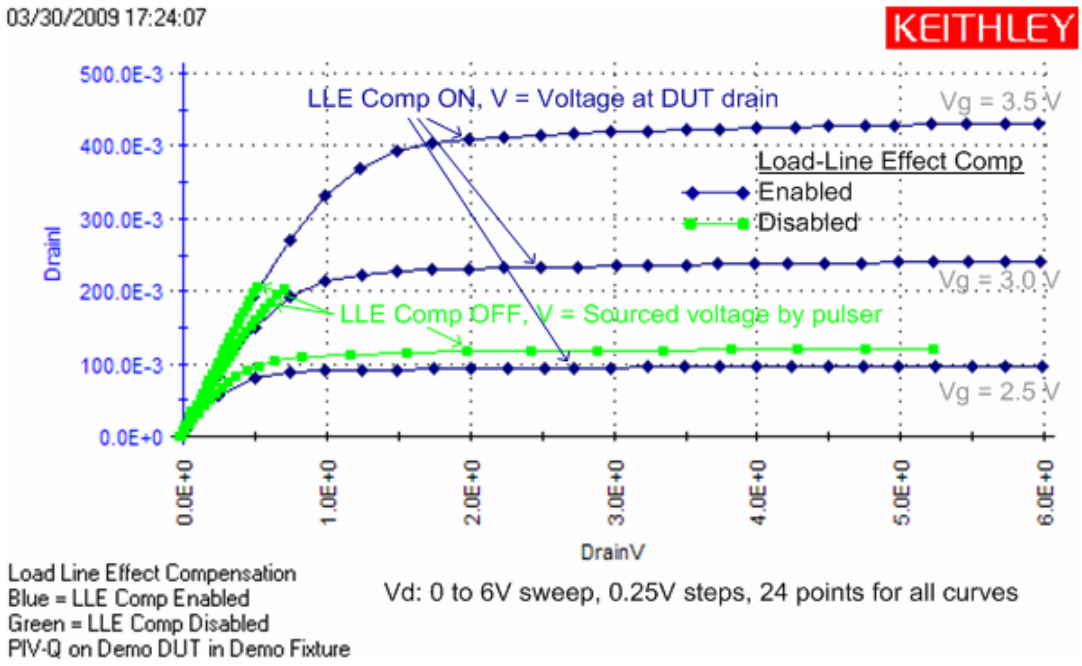
There are two parameters that control how the LLEC algorithm functions: Maximum number of iterations and tolerance window. For ITMs, the maximum number of iterations is fixed at 20 and the tolerance window is 0.3 percent. For UTMs, use the setmode function. The LLEC algorithm will iterate, trying to reach the target voltage until one of the following occurs: 1) The target voltage is reached (within tolerance specified) or 2) maximum number of iterations is reached. The maximum number of iterations must be equal for each channel in the test.

LLEC maintains even voltage spacing

Another advantage of using LLEC is that it maintains even voltage spacing during the test. For example, if the pulse sweep uses 250 mV steps, DUT voltage and current measurements will be performed at every 250 mV step. Data that is generated using even voltage spacing is ready to be fed into a mathematical model.

When not using LLEC, uneven voltage spacing may result due to load line effect. Figure 16-66 shows load line effect on a FET family of curves. The blue curves were generated with LLEC enabled and the green curves were generated with LLEC disabled. The Vg was been increased for the green curves to provide separation between the curves.

Figure 16-66
Load line effect on FET family of curves



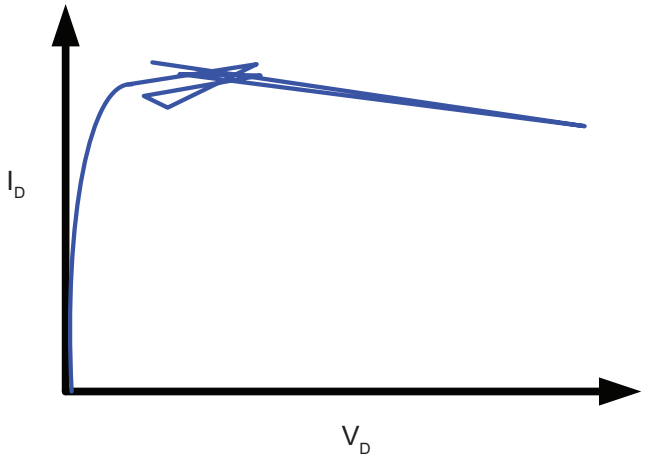
In Figure 16-66, each blue curve (LLEC on) is the result of a sweep from 0 to 6 V using 250 mV steps. Notice that the 24 pulse-measure points are evenly spaced.

The same sweep is used to generate the green curves (LLEC off). The best green curve is the one at the bottom (bias Vg = 2.5V). However, load line effect prevents the PMU from sourcing 6 V to the DUT and the 24 pulse-measure points are not evenly spaced. Modifying the sweep to 0 to 6.5 V will ensure that at least 6 V is output to the DUT, but voltage spacing will still be uneven. The green curves for the other two bias voltages (Vg = 3.0 V and 3.5 V) are even more adversely affected by load line effect.

Test considerations

- The magnitude of the pulse steps affects overall test time. Wider pulses, a higher number of pulses, and larger voltage steps at each sweep point, all increase the amount of time required for the LLEC algorithm at each sweep point, which lengthens the overall test time.
- There may be some high gain devices that will not test properly with LLEC enabled. In this case, you can disable LLEC. To disable LLEC in ITMs, see Controlling LLEC from an ITM. For UTMs, see the pulse_meas_sm and pulse_meas_wfm functions (in Section 8).

Figure 16-67
Curve showing poor LLEC compensation



LPT functions used to configure LLEC

There are four LPT functions used to configure LLEC for the PMU:

- pulse_load: Use this function to set the output impedance for the DUT when LLEC is disabled. Setting the DUT resistance is useful when the DUT resistance is known and is relatively constant
- setmode: Use this function to set the number of iterations for the LLEC algorithm or the tolerance window that determines if load line effect compensation is reached. The tolerance window is expressed as a percentage of the target voltage. The maximum number of iterations sets the maximum number of iterations that will be attempted by the LLEC algorithm. If the algorithm does not reach the target window, the measurements from last attempt will be returned.
- pulse_meas_sm and pulse_meas_wfm: Use these functions to enable or disable LLEC.

Controlling LLEC from an ITM

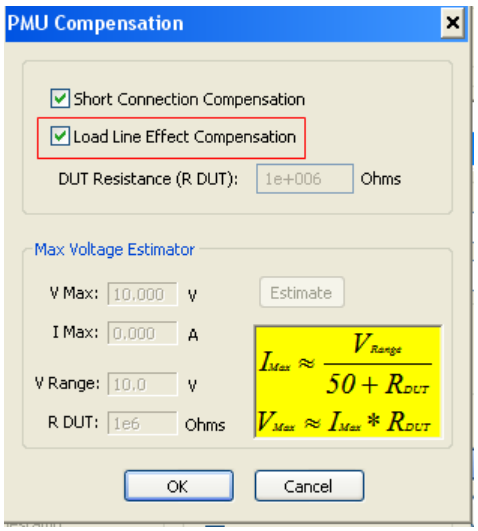
LLEC is enabled, on a per-channel basis, from the PMU Compensation window, which is opened as follows:

1. From the definition tab for an ITM, click the **FORCE / MEASURE** button (see Figure 16-22) for a PMU to open the FORCING FUNCTIONS / MEASURE OPTIONS window (see Figure 16-27).
2. Click the **Compensation** button to open the PMU Compensation window (see Figure 16-68).

Enabling LLEC

When LLEC is enabled from an ITM, the maximum iterations is set to 20 and the acceptance window is set to 0.003 (0.3 percent). To enable LLEC, insert a check mark in the box for **Load Line Effect Compensation** (as shown in Figure 16-68) and click **OK**.

Figure 16-68
Enabling LLEC

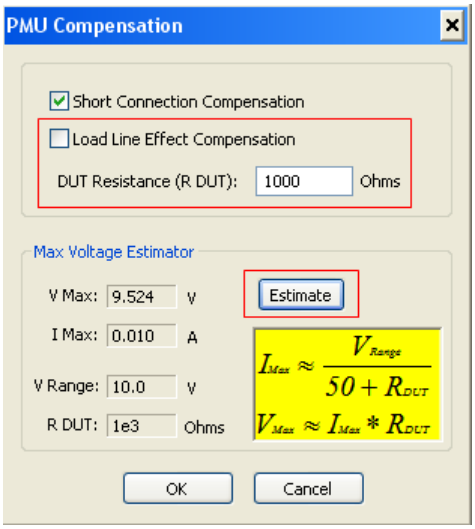


Disabling LLEC

With LLEC disabled (unchecked), you can input the known resistance of the DUT. The resistance value is then used in a compensation process to output the desired voltage. Perform the following steps to disable LLEC and set the output impedance:

1. To disable LLEC, click the **Load Line Effect Compensation** box to remove the check mark (see Figure 16-69).

Figure 16-69
Disabling LLEC



2. Enter the DUT resistance value in the **Ohms** field.
3. Click the **Estimate** button.

   The maximum voltage estimator is a tool that enables the user to easily calculate the maximum voltage and current based on the selected voltage range and the entered DUT resistance value. The estimator does not affect the pulse output.

   For a more accurate V Max estimate, add the interconnect resistance to the DUT resistance value. The typical resistance of a white SMA cable is 0.75 Ω, and the typical prober pin-to-pad resistance is 1 Ω to 3 Ω. Poorer pin-to-pad contacts could be in the range from 10 Ω to 15 Ω.

   The V Max and I Max values hold true with LLEC enabled or disabled. It is the maximum output of the PMU. In other words, the LLEC does not change the maximum output voltage or current of the PMU.
4. Click the **OK** button.

NOTE *The maximum available voltage is not affected by the settings for LLEC or DUT resistance. For example, the 10 V range can output 10 V into a high-impedance DUT (1 MΩ) and a lower voltage into lower impedance DUTs. For more information, see DUT resistance determines pulse voltage across DUT in Section 11.*

Understanding pulse shape effects

PMU minimum settling times versus current measure range

Similar to an SMU, the PMU and RPM current measure ranges require time to reach a settled value. The settle time required increases for the lower current measure ranges.

The pulse timing parameters (pulse width, rise, and fall) can be programmed to any valid values and are independent from the recommended minimum timing values required to obtain settled readings. If the chosen pulse width is too narrow, the lower current measure ranges are not available. The timing of the pulse top and pulse base (see Figure 16-4) are used, along with the minimum timing in the chart to determine which current measure ranges are available.

In ITMs, there is a reminder of the relationship between the current measure ranges and the minimum pulse width and transition times (see Settling Times NOTE in Figure 16-23). While modifying the PMU Timing dialog parameters, whenever you make a pulse timing change that affects the measure range of a PMU channel, a message noting the unavailable ranges is displayed (see Figure 16-46). You should check the recommended pulse width and period for the measure ranges noted in the message. When setting the PMU force measure options, if a timing parameter prevents use of one or more PMU or RPM measure ranges, the note in the center of the dialog turns red and lists the unavailable ranges (see Figure 16-27). Access the PMU force measure options by clicking the FORCE MEASURE button

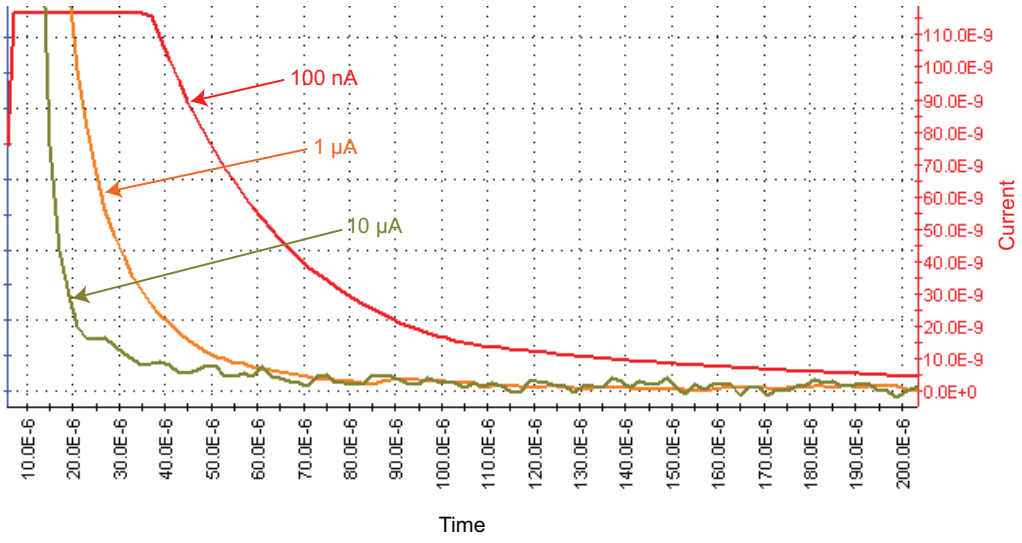
Figure 16-45 shows the recommended minimum pulse widths and transition (rise and fall) times for each current measurement range. These times are the minimum time necessary for the PMU, or PMU with RPM, to reach a settled value (into an open). Additional time must be entered to account for DUT settling time, usually due to RC effects.

If autoranging with the RPM is desired, note that the appropriate minimum timing values are the first row in the table. Note that the pulse timing values are not altered during the test, unless a time sweep or time step is configured. This means that the pulse width is not changed as the measure range changes. When using autoranging or limited autoranging, choose the recommended minimum timing values for the lowest chosen measure range.

It is possible to over-ride the settled measurement requirement for the PMU or PMU+RPM measurements by enabling a setting in KITE Tools | Options for the PMU (see PMU: Allow unsettled measurements:).

Figure 16-70 contains a graph of the current signal settling time of three current measure ranges of the Model 4225-RPM. The graph shows the settling time of a 1 V amplitude pulse in to an open (high impedance) with a 300 µs pulse width and a 1 µs transition time. Note that this figure does not show the voltage signal.

Figure 16-70
Model 4225-RPM current signal settling time example

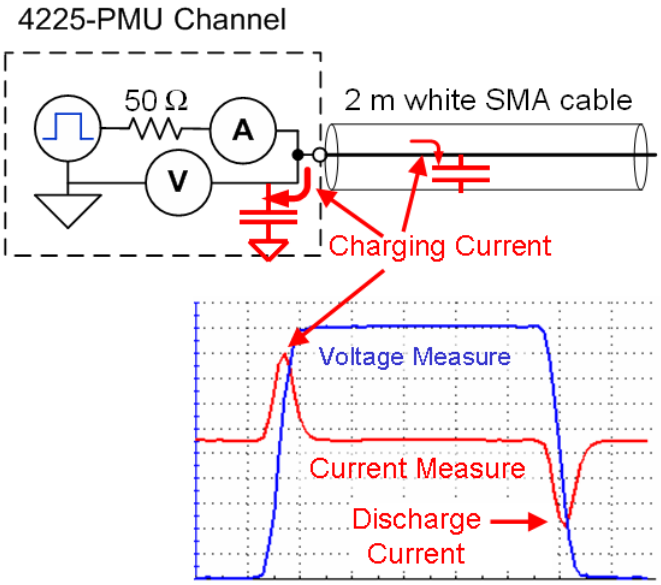


PMU capacitive charging/discharging effects

During pulse transitions, the measured current charges and discharges the capacitance in the system (see Figure 16-71, red waveform). This system capacitance consists of the cable capacitance, PMU (with RPM, if connected) capacitance, and device capacitance. Figure 16-71

shows the pulse waveform showing capacitive charging and discharging current waveform in relation to the applied voltage waveform of the PMU connected to the supplied 2 m (6.5 ft) SMA cable (no DUT connected).

Figure 16-71
Capacitive charging and discharging



The setup used to generate these waveforms is shown in Figure 16-71, and also shows the capacitance and the charging effect (red arrows) seen during pulse transitions. This setup shows a single channel of a PMU, with the supplied 2 m (6.5 ft) white SMA cable connected to the channel output. Note that the other end of the SMA cable is open (no connection).

The current shown in Figure 16-71 is measured by the PMU, but is not flowing through a device under test (DUT). The measured current is the sum of this charging or discharging current, as well as the current flowing through the DUT. This current is primarily caused by the capacitance in the cable and is described by the following equation:

$I = C * dV/dt$

Where:

I is the measured current
C is the capacitance
dV/dt is the pulse voltage amplitude divided by the rise (or fall) time

The equation shows that this effect is a function of the capacitance, as well as the dV/dt. Therefore, minimizing the capacitance will reduce this measurement artifact. The cabling is typically the largest contributor to the system capacitance. Slowing down the pulse transitions will also reduce the height of the current charging effect.

This capacitive charging current is primarily a measurement artifact, as the current does not flow through the DUT. Note that if a spot mean is taken during the settled portion of the pulse, then this charging does affect the spot mean measurement.

Figure 16-72 and Figure 16-73 shows the waveforms and setup for a pulse test on a resistor DUT and illustrates a configuration to eliminate this artifact. Figure 16-72 shows that the channel 2 current waveform does not have this current charging artifact. This is because channel 2 is not pulsing, so dV/dt = 0. Using channel 2 in this configuration is sometimes called “low-side measurement.” This measurement approach is useful when analysis of the current signal pulse transitions is required.

Figure 16-72
Low-side measurement waveforms

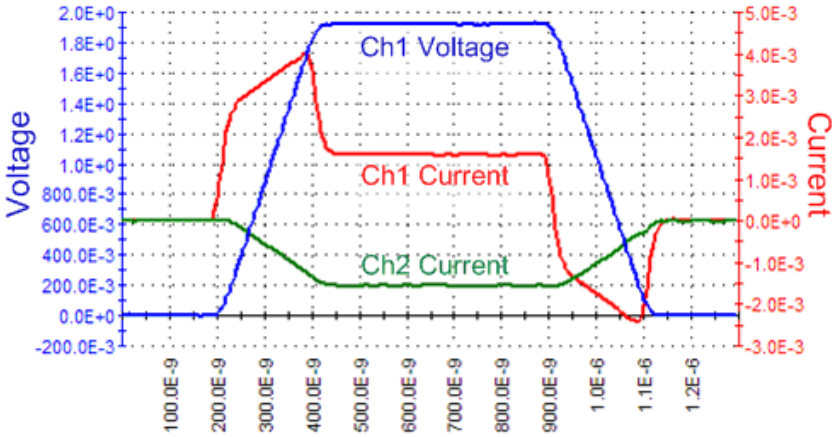


Figure 16-73
Setup for low-side measurement

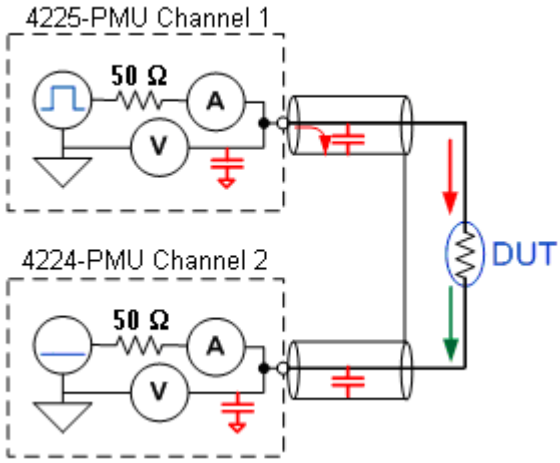


Figure 16-72 shows the current waveforms for both PMU channel 1 (high side) and channel 2 (low side) current measurements. Note that channel 2 does not show the capacitive charging effects. Also, note that the channel 2 current measurement is negative because the current is flowing into channel 2.

In Figure 16-73, the voltage pulse is applied by channel 1; channel 2 does not pulse. Therefore, there is no dV/dt, and therefore there are no charging or discharging currents during the pulse transition. The red arrows show charging + DUT current for channel 1. The green arrow illustrates the DUT current only for channel 2.

PMU and RPM measure ranges are not source ranges

Unlike a source-measure unit (SMU), the PMU and RPM current measure ranges are measure ranges only, not source and measure ranges. For example, the SMU 10 mA measure range has a maximum source and measure value of ±10.5 mA, including the five percent overrange. The 10 mA measure range of the PMU 10 V range has a maximum measurement of about ±10 mA, but the full source capability of the 10 V source, which is ±200 mA. An alternate way to present this difference is that the SMU range has a source compliance equal to the measurement limit, but the PMU and RPM ranges have a source compliance larger than the measure range. Note that for the maximum PMU current measure ranges (200 mA for the 10 V range, 800 mA for the 40 V range), the source limit is the same as the measure limit, so the 200 mA and 800 mA ranges act similar to the SMU current range.

This measure-only limit is necessary for the best performance of the pulse when using the PMU alone or with the RPM. Generally, the purpose of a pulse measurement is to minimize the time required to make a measurement in order to minimize device self-heating or some other time-based device behavior.

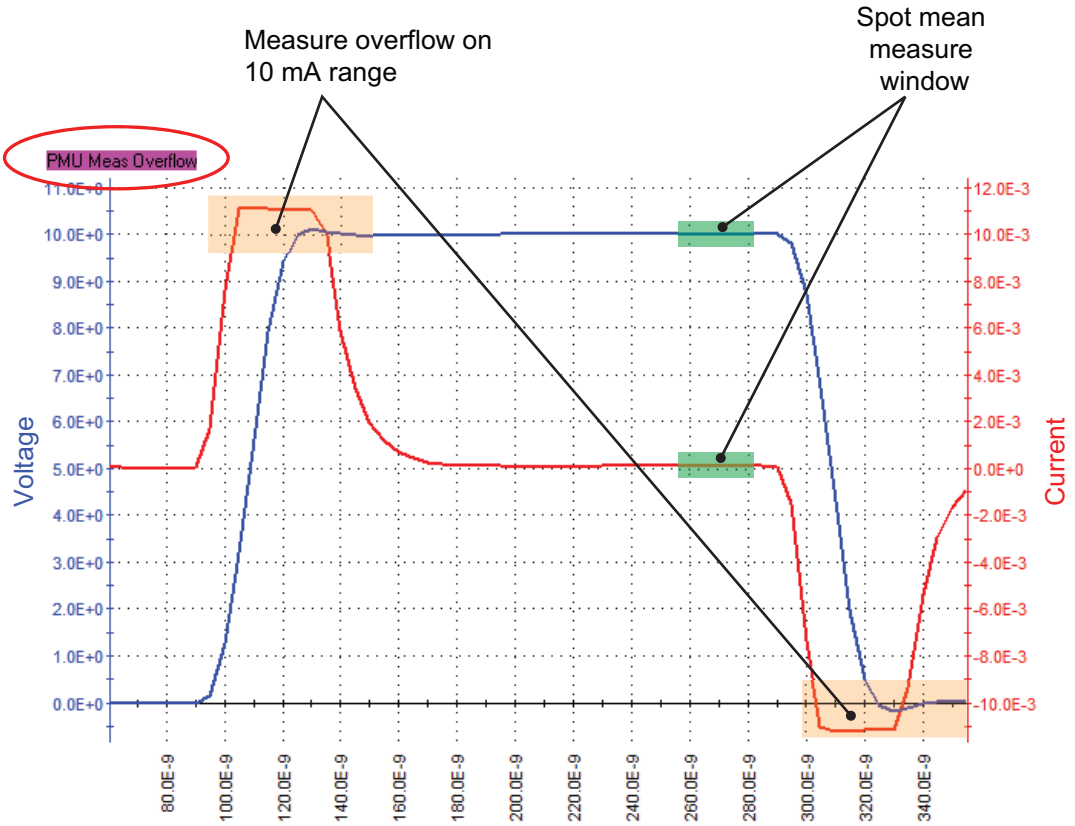
To minimize the measurement time, the signal at the device under test (DUT) must get to the desired voltage level and settle as quickly as possible. A key aspect of this goal is handling the capacitive charging effects during the pulse transitions. See “PMU capacitive charging/discharging effects” on page 16-66 for more information.

Since the interconnect and the DUT will always have some capacitance, it is best to charge up this capacitance as quickly as possible. This can be done by allowing the maximum amount of current to flow into the capacitor during charging. This may cause an overflow for the measure range during transitions, especially on the lower RPM measure ranges. Note that the measurement range cannot be changed within a pulse, so a single measure range must be used for the entire pulse. Figure 16-74 shows an example of measurement overflow on the 10 mA range of the RPM, pulsing into a 1 MΩ DUT. The capacitive charging current is not limited to 10 mA, but does cause a measurement overflow on the 10 mA range. Note the overflow warning in the upper left part of the graph (in the magenta box) and that maximum value for the 10 mA measure range is a bit larger, so Figure 16-74 show the current signal (red curve) clipped at about 11 mA.

If the PMU or RPM was current limited to the measure range, the charging rate would require a longer time to reach a settled signal. With the maximum current available at all measure ranges, the signal will settle as fast as possible, allowing for a good, DC-like, current measurement using the shortest possible pulse. This is especially important at the lower current measure ranges, when the settled part of the pulse may have a signal level in the nA (or single μA) value, but the charging current is possibly tens of μA (or mA, respectively).

See PMU capacitive charging/discharging effects for information on the cause of the capacitive charging effect and how to work around it.

Figure 16-74
PMU Waveform Capture (measurement overflow on the 10 mA current range)



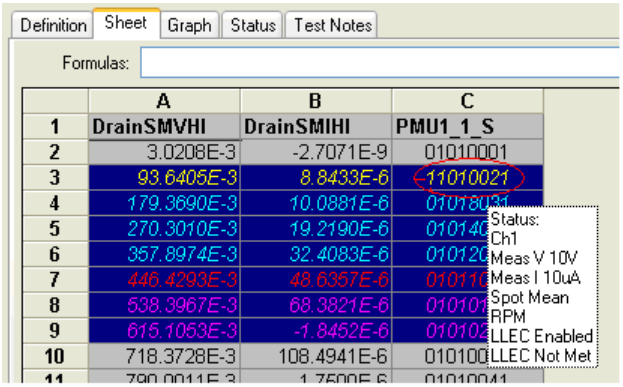
PMU measurement status

ITMs can provide status information for the Model 4225-PMU measurements in the Sheet tab of KITE (refer to "Miscellaneous measurement settings" on page 16-41). The data column for the status codes is labeled PMU*x*_*y*_S, where *x* is the PMU instrument number (PMU1, PMUs, and so on), and *y* is channel number (channel 1 or 2). The PMU status code indicates pulse measurement status, source and measure ranges, whether an RPM is connected, and LLEC status (see "Load line effect compensation (LLEC) for the PMU" on page 16-60), and flags any faults (errors).

When a pulse measurement fault occurs, the entire row of data related to the measurement will be highlighted in blue (see Figure 16-75). A single measurement may have more than one condition. The conditions display when the mouse pointer hovers over the status value in the sheet (see Figure 16-75). The data values in the flagged data row will be color-coded to identify the fault type as follows:

Red = Source in compliance
Magenta = Measurement overflow
Yellow = Load line effect compensation (LLEC) failed
Blue = Current, voltage, or power threshold reached

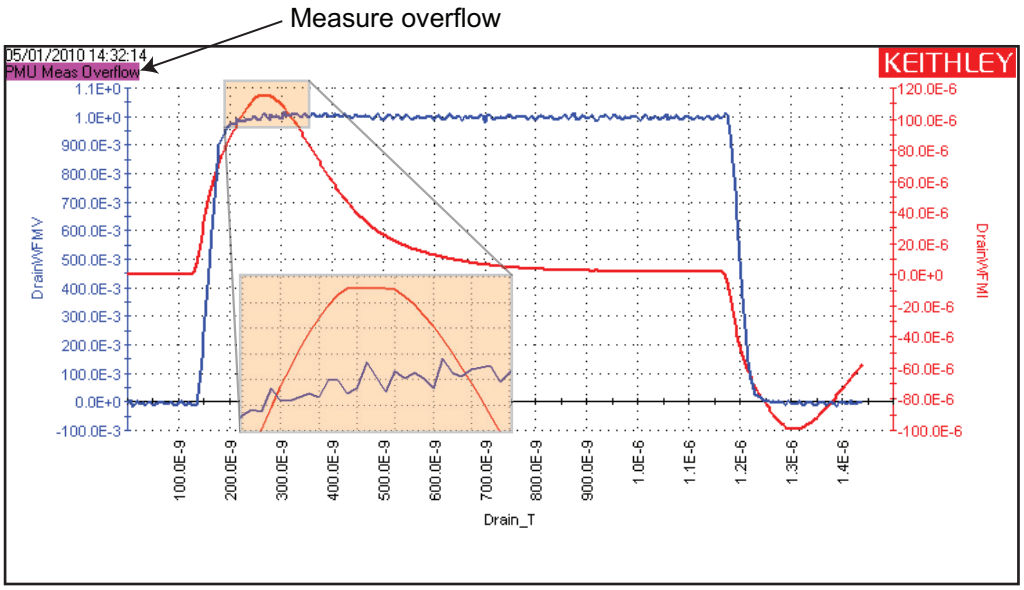
Figure 16-75
Status tab showing faults (example)



Placing the cursor on a flagged PMU1_1_S cell will open a window that summarizes the fault.

When a measurement fault occurs, a message appears in the upper left corner of the graph. Figure 16-74 shows a measurement overflow on the Graph. Figure 16-76 shows a graph with a PMU Measurement Overflow condition. When troubleshooting a measurement fault (such as the shown measurement overflow), use the measurement status code to determine the situation and possible fixes. See “Basic troubleshooting procedure” on page 16-75 for more information.

Figure 16-76
Sample measurement overflow



Status codes

As shown in Figure 16-75 each measurement includes an 8-digit status code. Assign the following letters to the code and use Table 16-14 to determine status:

A B C D E F G H

In Figure 16-75, the circled status code is 11010021:

A) 1 = LLEC failed, sweep not skipped (yellow indicates that LLEC failed)
B) 1 = RPM being used

C) 0 = Not applicable (always 0)
D) 1 = Spot mean measurement type
E) 0 = Source not in compliance, no threshold reached
F) 0 = No measurement overflows
G) 2 = 10 µA measure range
H) 1 = Channel 1, 10 V measure range

Table 16-14
PMU measurement status codes

Code letter	Summary or description	Value
A	Load line effect compensation (LLEC) and Sweep	0 = LLEC disabled, sweep not skipped 1 = LLEC failed, sweep not skipped 3 = LLEC successful, sweep not skipped 4 = LLEC disabled, sweep skipped 5 = LLEC failed, sweep skipped 7 = LLEC successful, sweep skipped
B	RPM mode settings	0 = No RPM 1 = RPM 2 = Bypass; PMU 3 = Bypass; SMU 4 = Bypass; CVU
C	Reserved for future use	Always 0
D	Measurement type	1 = Spot mean 2 = Waveform
E	Current threshold, voltage threshold, power threshold, and source compliance	0 = None 1 = Source compliance 2 = Current threshold reached or surpassed 4 = Voltage threshold reached or surpassed 8 = Power threshold reached or surpassed
F	Voltage measure overflow and current measure overflow	0 = No overflow 1 = Negative voltage overflow 2 = Positive voltage overflow 4 = Negative current overflow 5 = Negative voltage and negative current overflow 6 = Positive voltage and negative current overflow 8 = Positive current overflow 9 = Negative voltage and positive current overflow A = Positive voltage and positive current overflow
G	Current measure range	0 = 100 nA (RPM only) 1 = 1 µA (RPM only) 2 = 10 µA (RPM only) 3 = 100 µA 4 = 1 mA (RPM only) 5 = 10 mA 6 = 200 mA 7 = 800 mA
H	Channel number and voltage measure range	1 = Ch1, 10 V 2 = Ch2, 10 V 5 = Ch1, 40 V 6 = Ch2, 40V

Model 4220-PGU and Model 4225-PMU output limitations

In addition to the maximum output of the PMU and PGU instrument cards (see "DUT resistance determines pulse voltage across DUT" on page 11-15), the pulse instrument cards also have a limit for the number of large amplitude pulse transitions within a period of time. The Model 4200-SCS system enforces limits on the quantity and amplitude of waveforms that the Model 4220-PGU and Model 4225-PMU may generate. A test exceeding these internal limits will generate Error code -830 (see Table 8-19). To fix this problem, increase pulse period, decrease voltage amplitude, or both.

Model 4200-SCS power supply limitations

In some system configurations, the Model 4200-SCS power supply cannot supply enough current if a test has too many high power instruments enabled. Some system configurations may have enough instruments installed to exceed the power supply limit if the desired test has too many channels enabled.

The KITE software tracks the instruments used in a test and calculates the maximum power required and compares it to the maximum available power. If the test requires too much power, a message (Figure 16-77 or Figure 16-78) is displayed and the test will not run.

Table 16-15 and the equations below it show how the power is calculated. The maximum power available for each instrument in the test module is used in the calculation. The High Power SMU (4210-SMU), Ultra-Fast I-V Module 4225-PMU and Dual Pulse Generator 4220-PGU draw the majority of the power in the 4200-SCS chassis.

There are two parts to the total power supply draw. The first part is the power required for the instruments while the 4200 is idle (turned on, but not testing). The second part is the power required by the instruments taking part in the test. Note that medium power SMUs (4200-SMU), 4200 PreAmps (4200-PA) or 4210-CVU cards are not included in the equations, as their power draw is not significant.

Table 16-15
Model 4200-SCS power requirements

Instrument	Idle power	Test power
High Power SMU (4210-SMU)	NS	45
4225-PGU	29.4	NA
4225-PMU	50.4	NA
4225-RPM	4.2	NA
10V PGU or PMU channel*	NA	8.4
40V PGU or PMU channel*	NA	54.6
Medium Power SMU (4200-SMU)	NS	NS
4200-CVU or 4210-CVU	NS	NS

NS = Not Significant
NA = Not Available
* There are two channels per PGU and PMU instrument card.

Equations to calculate power:

$Power_{IDLE} = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)]$
$Power_{TEST} = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)]$
$Power_{TOTAL} = Power_{IDLE} + Power_{TEST}$

$Power_{TOTAL}$ = 500 maximum. If $Power_{TOTAL}$ is less than or equal to 500, then the test proceeds.

Where:

nPGU = number of 4220-PGU cards in the 4200-SCS chassis (idle power draw)
nPMU = number of 4225-PMU cards in the 4200-SCS chassis (idle power draw)
nRPM = number of 4225-RPM modules connected to PMUs (idle power draw)
nHPSMU = number of High Power SMU (4210-SMU) in the test
nC10 = number of 10Volt PGU or PMU channels in the test
nC40 = number of 40Volt PGU or PMU channels in the test

Example 1: 4200-SCS with two 4210-SMU, four 4225-PMU and eight 4225-RPMs. The test uses all eight PMU+RPM channels set to the 10V range (no SMUs in test).

$Power_{IDLE}$ = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)] = [(29.4 * 0) + (50.4 * 4) + (4.2 * 8)] = 201.6 + 33.6 = 235.2

$Power_{TEST}$ = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)] = [(45 * 0) + (8.4 * 8) + (54.6 * 0)] = 67.2

$Power_{TOTAL} = Power_{IDLE} + Power_{TEST}$ = 235.2 + 67.2 = 302.4

This test has $Power_{TOTAL} \leq 500$, so this test will proceed.

Example 2: 4200-SCS with two 4210-SMU, four 4225-PMU and 8 4225-RPMs. The test uses five PMU+RPM channels set to the 40V range (no SMUs in test).

$Power_{IDLE}$ = [(29.4 * nPGU) + (50.4 * nPMU) + (4.2 * nRPM)] = [(29.4 * 0) + (50.4 * 4) + (4.2 * 8)] = 201.6 + 33.6 = 235.2

$Power_{TEST}$ = [(45 * nHPSMU) + (8.4 * nC10) + (54.6 * nC40)] = [(45 * 0) + (8.4 * 0) + (54.6 * 5)] = 273

$Power_{TOTAL} = Power_{IDLE} + Power_{TEST}$ = 235.2 + 273 = 508.2

This test has $Power_{TOTAL} > 500$, so this test will not proceed. Reduce the number of 40V channels from 5

Table 16-16 shows the 4200-SCS Power Requirements for valid combinations for the 4225-PMU, 4225-RPM, and High Power SMU.

Table 16-16
4200-SCS power requirements for valid combinations of internal system instruments

Idle power		Power used in test					
nPMU	nRPM	nC10	nC40	nHPSMU	$Power_{IDLE}$	$Power_{TEST}$	$Power_{TOTAL}$
2	4	0	4	0	117.6	218.4	336
2	4	0	4	1	117.6	263.4	381
2	4	0	4	2	117.6	308.4	426
2	4	4	0	2	117.6	123.6	241.2
3	6	6	0	0	176.4	50.4	226.8
3	6	6	0	1	176.4	95.4	271.8
3	6	6	0	2	176.4	140.4	316.8
3	6	0	5	0	176.4	273	449.4
3	6	0	5	1	176.4	318	494.4
4	8	8	0	0	235.2	67.2	302.4
4	8	8	0	1	235.2	112.2	347.4
4	8	8	0	2	235.2	157.2	392.4
4	8	0	4	0	235.2	218.4	453.6
4	8	0	4	1	235.2	263.4	498.6
5	10	10	0	0	294	84	378
5	10	10	0	1	294	129	423
5	10	10	0	2	294	174	468

Table 16-16
4200-SCS power requirements for valid combinations of internal system instruments

Idle power		Power used in test					
nPMU	nRPM	nC10	nC40	nHPSMU	$Power_{IDLE}$	$Power_{TEST}$	$Power_{TOTAL}$
5	10	0	3	0	294	163.8	457.8
5	10	0	2	1	294	154.2	448.2
5	10	0	2	2	294	199.2	493.2
6	12	12	0	0	352.8	218.4	336
6	12	12	0	1	352.8	263.4	381
6	12	0	2	0	352.8	308.4	426
6	12	0	1	1	352.8	123.6	241.2

The power limit check is performed in both ITMs and UTMs. In ITMs, exceeding the power limit will display a message similar to the one shown in Figure 16-77 when configuring PMU channels in the Definition tab or PMU Force/Measure setting.

For UTMs, the message appears in the KITE message window (see Figure 16-78).

Figure 16-77
ITM maximum power exceeded message (example)

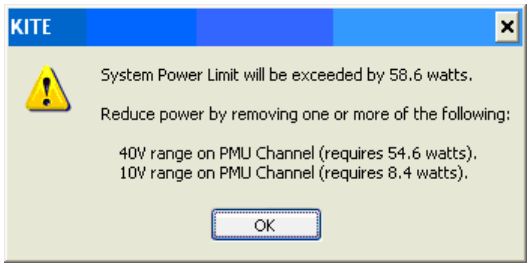
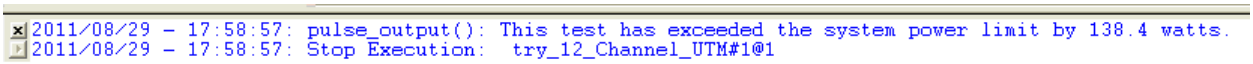


Figure 16-78
UTM maximum power exceeded message (example)



Basic troubleshooting procedure

If the ITM pulse I-V results do not meet expectations, use the following steps as a guide for troubleshooting. Because the pulse I-V results extract the spot mean measurements from the top of the pulse (see Pulse IV), good pulse I-V results require a reasonable pulse shape.

Step 1. Verify prober connections from the PMU or RPM to the DUT

A. Use cabling and connections optimized for high frequency (≥150 MHz).
B. Connect the low side of the device under test (DUT) to the shield of the coax cable (see Shield connections), or connect the low side to another PMU or RPM channel (Figure 16-17 or Figure 16-24).
C. Connect the shields together (refer to Figure 16-21).
D. If not using the supplied cabling, minimize the loop area (refer to Shield connections) created by the shield and center conductor. Do not use the GNDU connection for the return or ground path for any pulse I-V signal.
E. Minimize the cable length (see Cable length).

Step 2. Verify the pulse shape

Check that the pulse shape provides a flat, settled portion near the end of the pulse top.

A. Configure the test for Waveform Capture. See “Step 1) Select the test and measure modes” on page 16-30 for more information.

B. Ensure that voltage, current, and time waveforms are measured. See “Waveform capture measurement configuration” on page 16-41 for more information.

C. Enable the status for each PMU channel in the test. See “Miscellaneous measurement settings” on page 16-41 for more information.

D. Pick a voltage level for each channel that puts the device into the area of the curve that is questionable. If the a large portion of the pulse I-V curve is under question, then viewing the waveform shape at two or three different voltage levels may be necessary.

E. Configure the ranges to match the pulse I-V test. See “Step 3) Set the measure ranges” on page 16-35 for more information.

F. Configure the connection and LLE compensation. See “Controlling LLEC from an ITM” on page 16-64 for more information. To match the pulse I-V test conditions, enabling LLEC will allow the PMU to compensate for lower voltage levels at the test device when current is flowing.

G. Configure the graph with the time value on the x-axis, all voltage measurements on Y1, and current measurements on Y2. On the graph, click on graph properties button to see the graph definition dialog (see Figure 16-74). By default, the voltage waveforms are blue and use the left (Y1) axis; the current waveforms are red and use the right (Y2) axis.

H. Save the project

I. Run the test and view the waveform on the graph. While viewing the graph, check that the voltage has a fairly flat top, without significant ringing or oscillations.

- There may be current peaks during the pulse transitions (see PMU capacitive charging/discharging effects). The peaks during the transitions are expected. The current waveform may show settling (similar to Figure 16-70), but should not have significant ringing. An example of a good waveform shape is Figure 16-74.
- Due to interconnect and DUT settling time requirements, low current measurements (< ~10 μA) may require a wider pulse than the recommended minimum timing values (see PMU minimum settling times versus current measure range). This is especially important for < 1 μA level current measurements using the Model 4225-RPM.
- A waveform graph may show a Measurement Overflow condition.
  - In Figure 16-74, this overflow is due to the current flowing because of the capacitive charge and discharge effects (see PMU capacitive charging/discharging effects).
  - This overflow during the pulse transitions does not effect the spot mean measurements (see Pulse IV, because the spot mean is taken during the settled portion of the pulse top (Figure 16-74). Note that this error would not occur during pulse I-V, because the spot mean measurement window does not include the rising or falling edges of the pulse. See “Measure Mode” on page 16-31 for more information.

Step 3. Is the pulse level correct for each channel?

A. If the pulse level is not correct for each channel, enable LLEC. See “Controlling LLEC from an ITM” on page 16-64 for more information. To compensate for the IR drop effect, the LLEC algorithm applies multiple pulses at each sweep step; make sure that the DUT behavior is not adversely affected by the LLEC approach.

B. Append the test and compare results. If the results match the test settings, then the issue was the lack of LLEC.

The load line effect can reduce the voltage level at the DUT terminal when current is flowing. See “Load line effect compensation (LLEC) for the PMU” on page 16-60 for more information.

Figure 16-80 shows a family of curves from a fairly high power device with the load line effect. Figure 16-81 shows the same device tested with load line effect compensation enabled. With LLEC enabled, the curves stop at the programmed V_d = 12 V. Note that the top curve, in the red circle, did not reach the 12 V setting. This is because the PMU 40 V source range reached source compliance (see the warning in the upper left of the graph). In this case, the PMU is at its limit and cannot source any more voltage or current in to this particular resistance. See the Mode 4200-SCS Technical Data sheet for more information on the PMU maximum source power vs. device resistance. The Model 4200-SCS Technical Data sheet can be found in the Complete Reference link on the Model 4200-SCS desktop, under the Data Sheets link.

Step 4. Is the pulse I-V curve suspect?

If the waveform has a good shape (Step 2. Verify the pulse shape), and the pulse level is correct (Step 3. Is the pulse level correct for each channel?), but the pulse I-V curve is suspect, perform the steps below.

NOTE *LLEC may not respond properly for a high-gain transistor (for example, a compound semiconductor-based amplifier or power transistor).*

A. Disable LLEC. See “Controlling LLEC from an ITM” on page 16-64 for more information.

B. Use other test parameters to provide boundaries for the test envelope (with LLEC disabled, the source voltage must be bounded):

   1) First, choose the maximum voltage for the selected source range. For example, many high power transistors require fairly high voltages and currents, so the PMU 40 V source range is common. See “Step 3) Set the measure ranges” on page 16-35 for more information.

   2) Set the thresholds (see the “Thresholds” bulleted item in the Miscellaneous measurement settings paragraph). Enable the Voltage threshold and enter the maximum voltage for the DUT. For a transistor, set this voltage for the maximum voltage for the drain.

   3) Enter the maximum power for the test device. An example of a transistor test with LLEC disabled with a power threshold of 3 W and a voltage threshold of 12 V is shown in Figure 16-82. Note that each threshold allows the test to be bounded. Also note that the thresholds do not stop the test at exactly the threshold value, but only after the threshold has been exceeded. One test point always exceeds the threshold. You can reduce the amount that the threshold is exceeded by using smaller sweep step sizes.

Figure 16-79
Four-channel waveform test graph definition (displaying V and I for each channel)

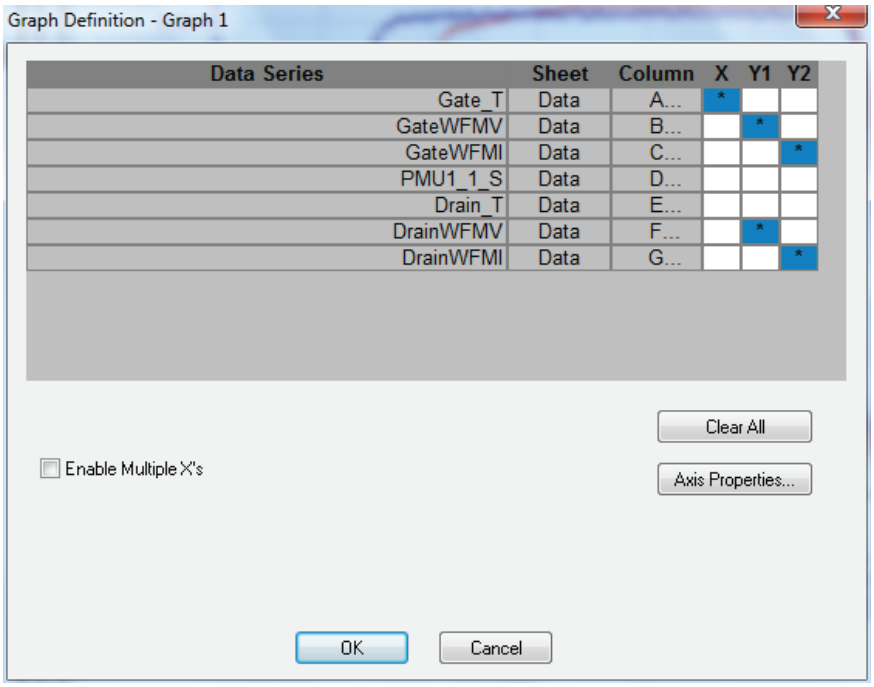


Figure 16-80
Vd-Id family of curves, showing load line effect (LLEC disabled)

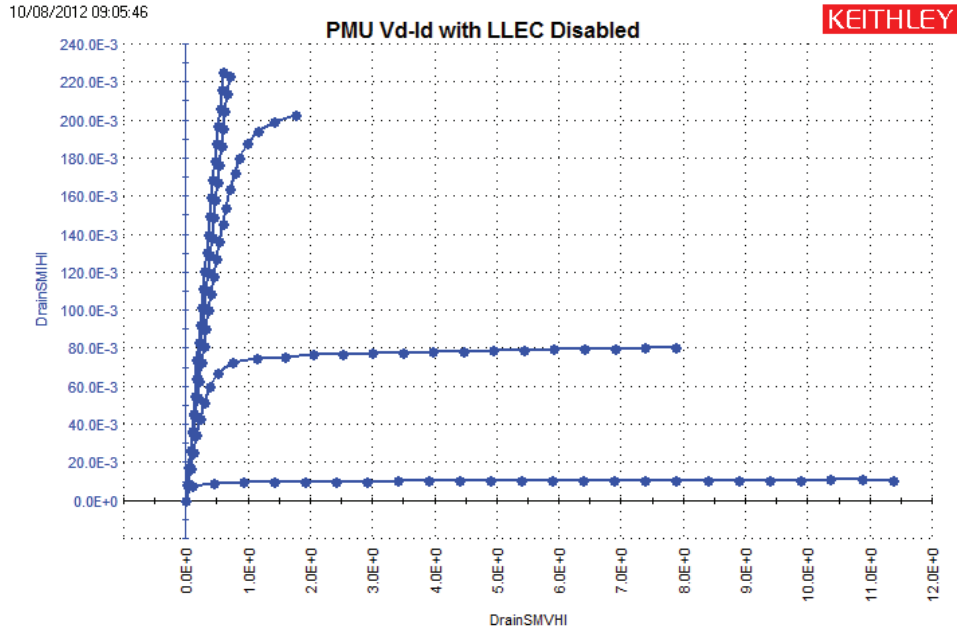


Figure 16-81
Vd-Id family of curves with LLEC enabled (every curve finishes at Vd = 12 V)

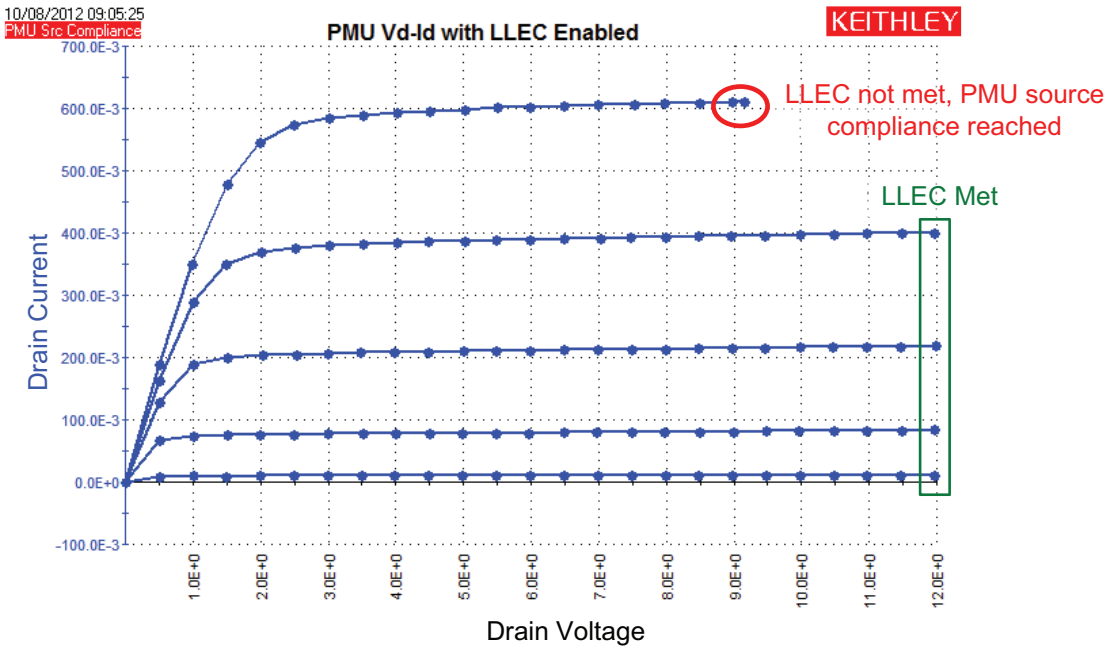
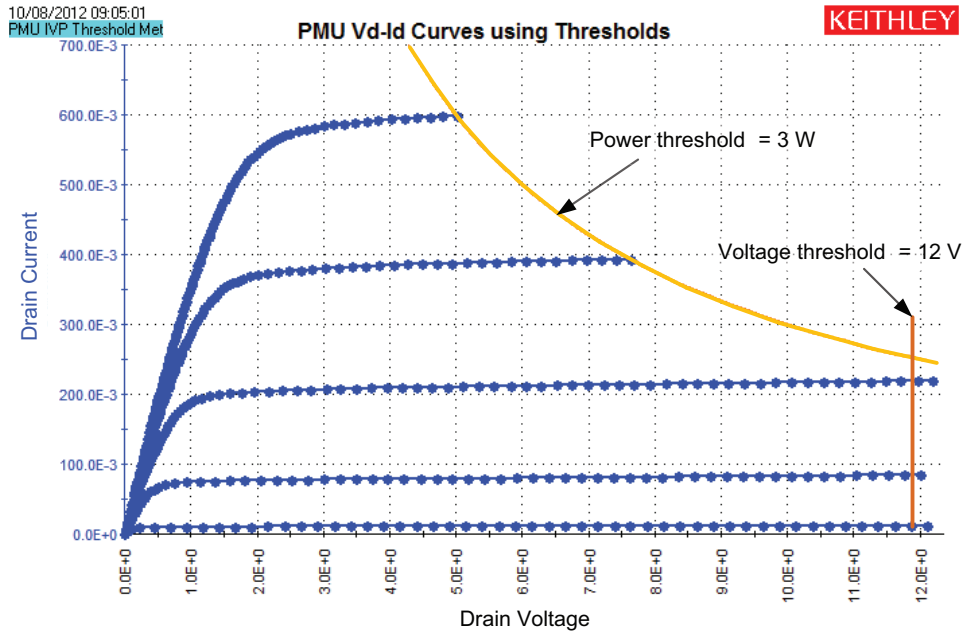


Figure 16-82
Vd-Id family of curves with LLEC disabled (thresholds: voltage = 12 V; power = 3 W)



For User Test Modules (UTMs), although the process described above applies, there are a few differences. Although ITMs and UTMs can perform similar tests, UTMs are based on user modules (see “Interactive Test Modules (ITMs) and User Test Modules (UTMs)” on page 6-14). UTMs have a much wider range of test behaviors and possibilities than ITMs. This document provides a

framework for obtaining information to assist in troubleshooting UTMs (individual UTM issues are not covered).

In addition to the above procedure based on unexpected results, UTM troubleshooting also involves error messages or codes. Typically the user modules are written for a specific test or requirement and have minimum error checking. This means that parameter values or combinations of parameter settings may cause an error, which is the primary feedback about the test status.

This error may be generated from within the user module, or by an LPT command. The red boxes in Figure 16-83 and Figure 16-84 show the user module description for PMU_IV_sweep_Example, in the UTM GUI and UTM Classic views, respectively. This description is part of the user module and may contain a description of the test, hardware requirements, individual parameter descriptions, and error code listings. The error code listings are after the parameter descriptions, which are typically at the bottom of the description content.

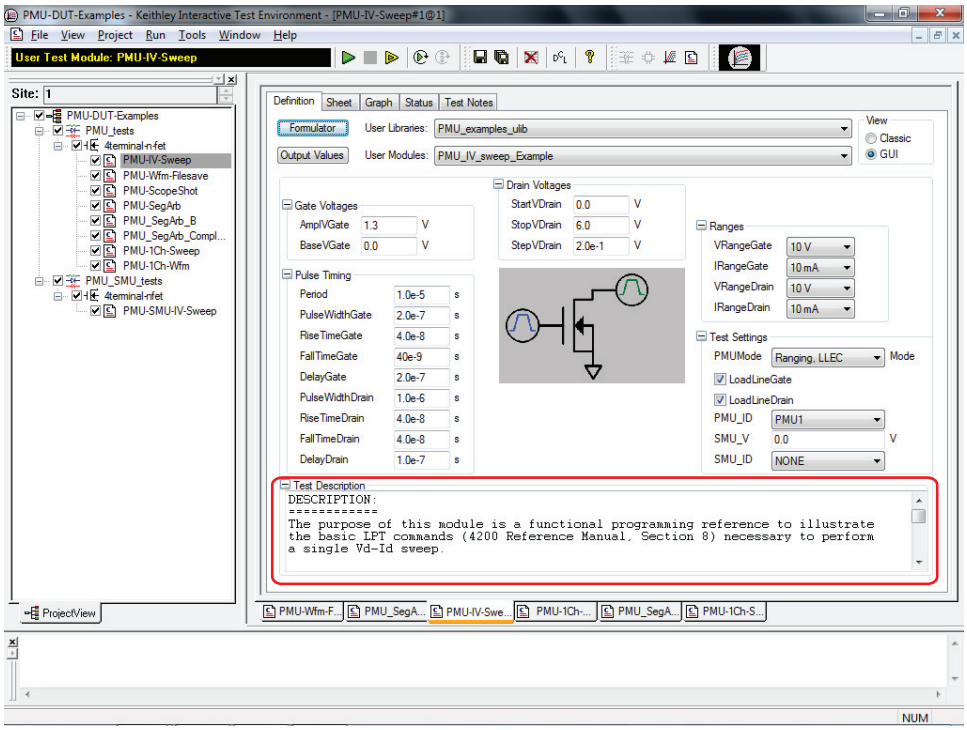
This user module allows for an optional SMU to DC bias a transistor bulk while performing a pulse I-V sweep with a Model 4225-PMU and optional Model 4225-RPMs. However, due to system restrictions, this SMU must not be connected to the test device through the Model 4225-RPMs (an error results).

This example test is configured to use SMU1. SMU1 is connected through RPM1 of the system. The following examples show either a UTM GUI or a UTM classic view of a PMU-IV-Sweep.

1. Some UTM GUI Views have tooltips for the parameters. Hovering the mouse pointer over the control or text entry field displays the tooltip. Figure 16-85 shows the tooltip for the SMU_ID of this user module. Note that is informs that the SMU must not be connected to an RPM or to a test device terminal that responds to pulsing.
2. Set the SMU_ID from NONE to SMU1 (Figure 16-86).
3. Click the **Append** button.
4. Look at the results in Figure 16-87. Note that the displayed curve was already there, the append button was clicked, so a second curve should have been displayed. But there is only 1 curve, so something went wrong. Looking at the message window at the bottom of the screen, there is a message "forcev(): Cannot force when not connected." You can also look at the Sheet which shows the results for the Append. Figure 16-88 shows the sheet for Append1. Note that there is an -233 error code in the first column. Generally, all user modules use the sheet's first column for the run status of the test. Negative numbers mean a problem and that the test did not run, or did not run properly. Note that the measurements in the other columns are also all zeros (0), indicating some type of problem. In this case the test did not run, but encountered an error during the initial instrument configuration.
5. In addition to the tooltip text, there may also be details in the test description for particular issues, and/or a listing of the errors. Figure 16-89 shows a portion of the test description using the UTM GUI View, while Figure 16-90 shows a different portion of the test description showing the error codes listing from the UTM Classic View.
6. Although using the user module test description should be the primary resource for troubleshooting non-performing tests, it may also be necessary to research further using the error codes. Three-digit negative error codes are most likely LPT command error codes. Section 8 has details about the LPT error codes "LPT Library Status and Error codes" on page 8-209 which may be useful in further troubleshooting. One-digit, four-digit, or five-digit error numbers are most like from the user module, so refer to the user module description for information about these type of errors.

NOTE *The following example is located in the _Pulse project directory.*

Figure 16-83
GUI view sweep UTM



NOTE *The following example is located in the _Pulse project directory.*

Figure 16-84
Classic view UTM sweep

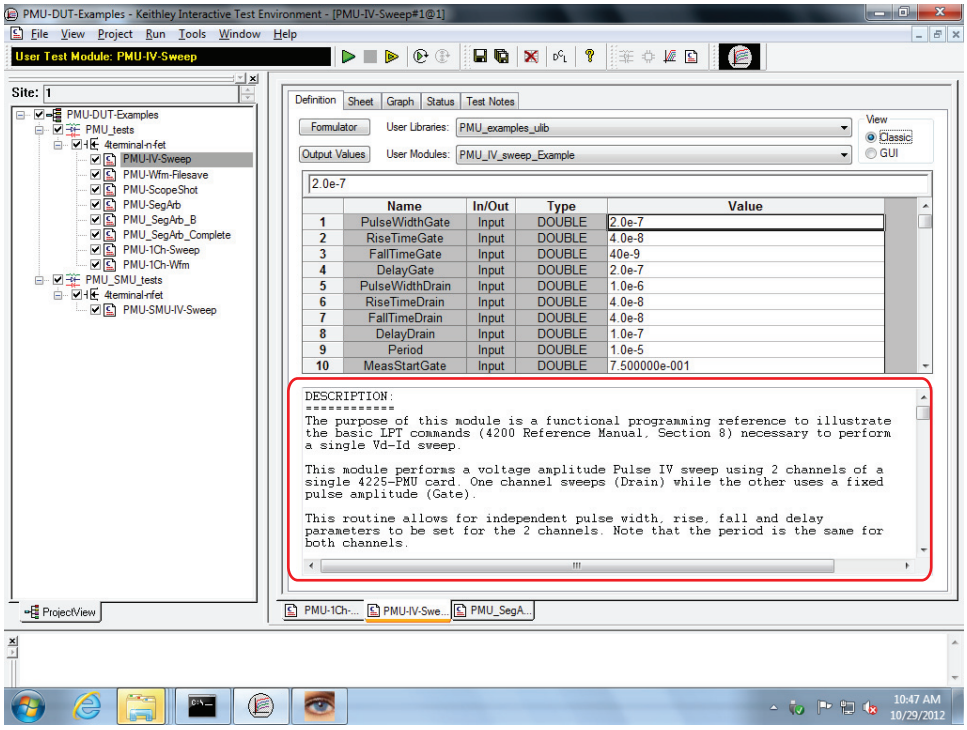


Figure 16-85
GUI view UTM sweep showing SMU_ID tooltip text

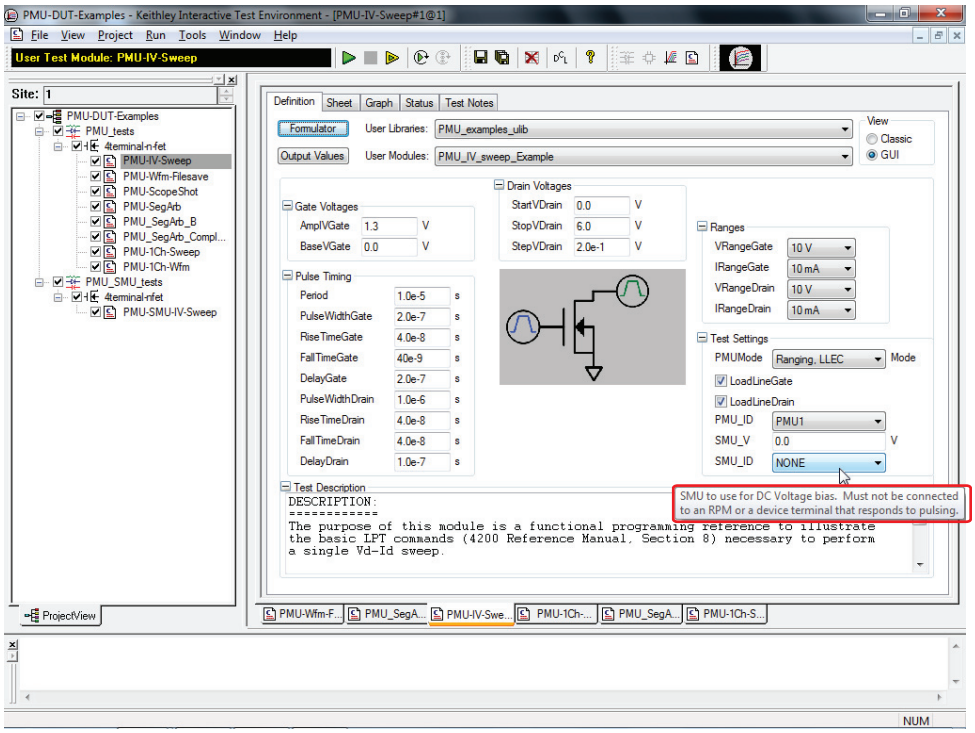


Figure 16-86
GUI view of UTM showing SMU_ID set to SMU1

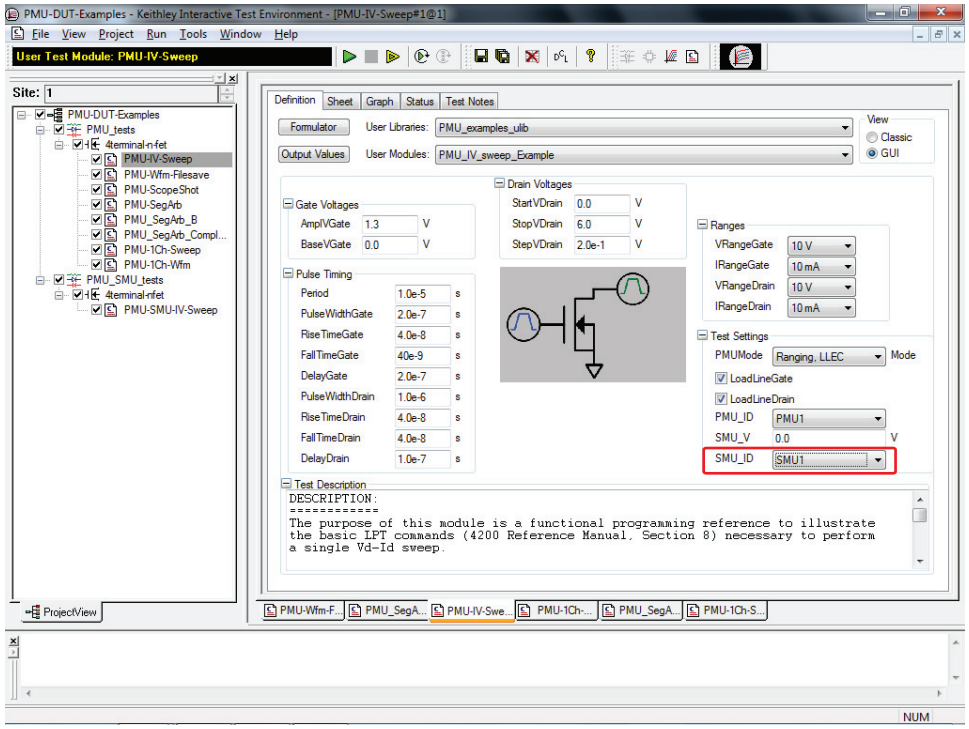


Figure 16-87
GUI view of UTM showing error message

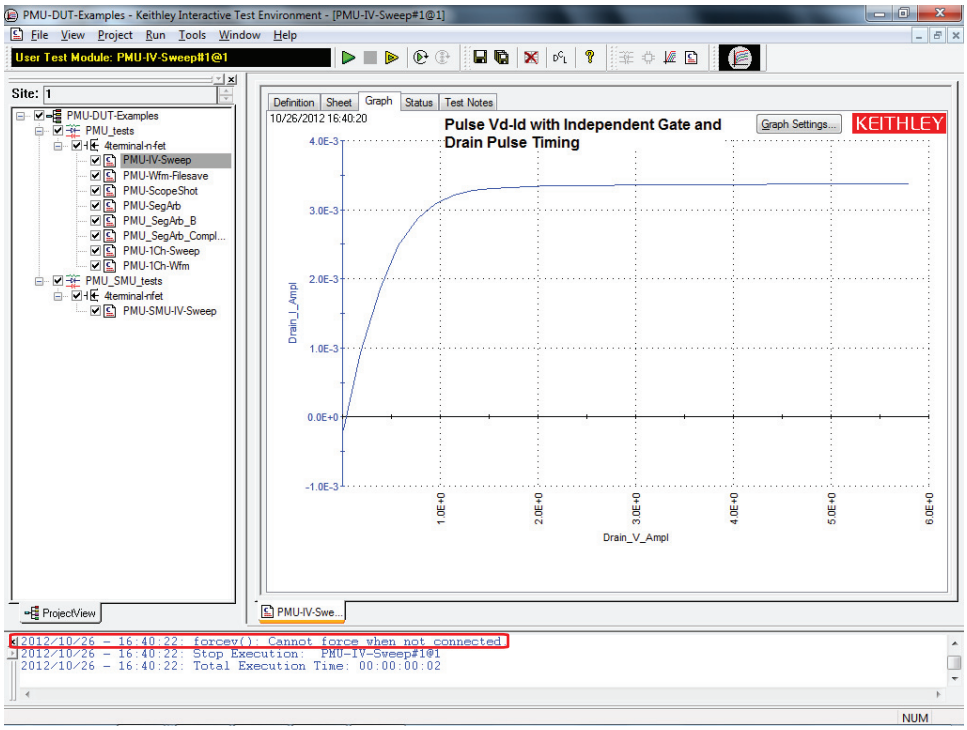


Figure 16-88
GUI view of sweep Append1 Sheet

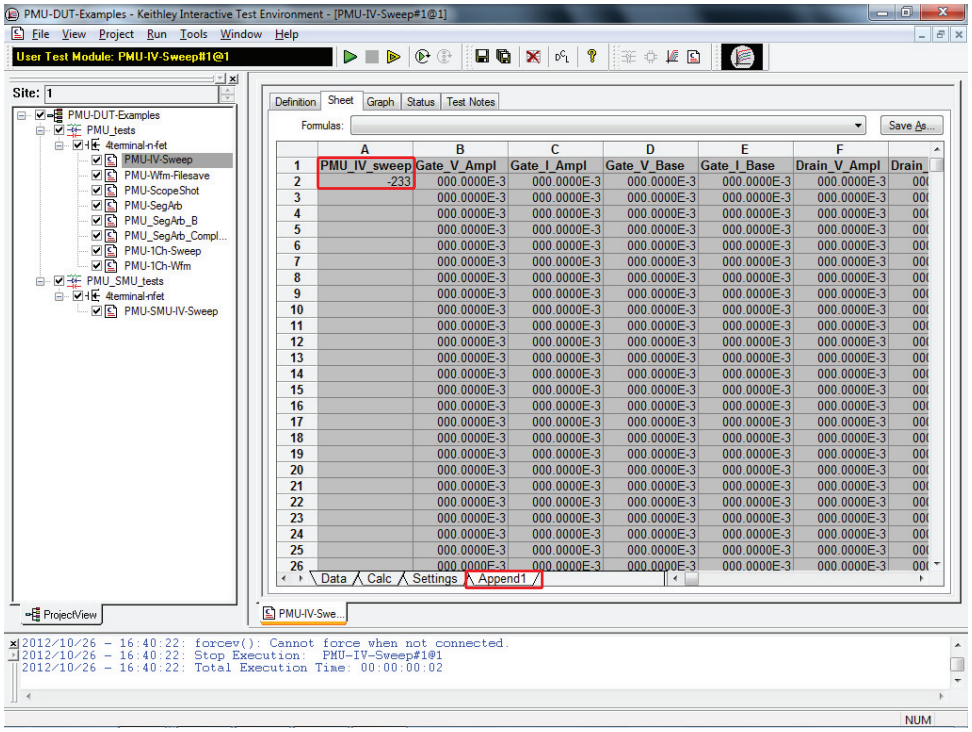
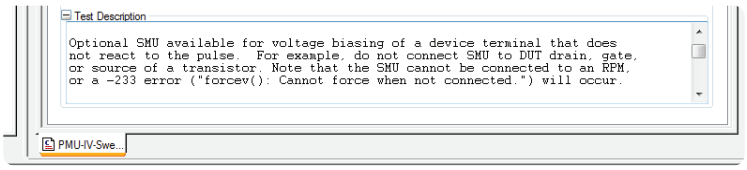
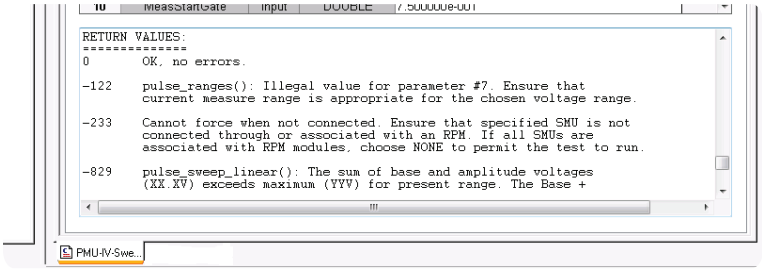


Figure 16-89
GUI view of UTM sweep showing SMU-related help



NOTE *The following figure illustrates a view containing a partial listing of the errors.*

Figure 16-90
Classic View of UTM sweep description



Pulse project plans summary

The project plans that use the pulse generator cards are summarized in Table 16-17.

Table 16-17
Pulse project plans

Project plan/test	Project folder	Pulse generators supported	Description
chargepumping See Project summary for test details	**_Pulse**	PGU, PMU, PG2	The measures the charge pumping current (I_{CP}) on a MOSFET. Includes UTMs for performing common charge pumping test methods, including base voltage sweeps, amplitude sweeps, rise time sweep, fall time sweep, linear sweep of the test frequency, and a log sweep of the test frequency.
default	**default**	PMU	Summarizes the ultra fast pulse IV tests that have been added to the default project plan. Included are pulsed and transient I-V testing for a MOSFET, a diode, a resistor, and a capacitor.
pulse-vds-id			This test generates a pulse IV drain family of curves on a MOSFET.
waveform-meas			This test performs a waveform capture of drain current versus drain voltage.
pulse-resistor			This test performs a pulse IV sweep of a resistor.
pulse-diode			This test performs a pulse IV sweep of a diode.
pulse-cap			This test generates a transient IV response of a capacitor.
PMU-DUT-Examples	**_Pulse**	PMU	This project contains example test modules to test a MOSFET. See PMU-DUT-Examples project for details on using this project.
PMU-IV-Sweep			This test performs a two-channel pulse IV test on a MOSFET. One channel of the PMU applies a pulse train to the gate, and the other channel performs an amplitude sweep for the drain, with independent pulse timings for each channel. A SMU can be used to provide voltage bias for the substrate.
PMU-Wfm-Filesave			This test performs a two-channel, two-level pulse waveform capture using Segment ARB pulse mode. This test allows for up to 1,000,000 waveform samples (per A/D) to be saved to a file. An SMU can be used to provide voltage bias for the substrate.
PMU-ScopeShot			This test performs a two-channel waveform capture with spot means. This test also allows adjustment of the spot mean measure window. A SMU can be used to provide voltage bias for the substrate.
PMU-SegArb			This test performs a two-channel waveform capture using the Segment ARB pulse mode. A SMU can be used to provide voltage bias for the substrate.
PMU-SegArb-B			This test is similar to the PMU-SegArb test, but allows for spot mean or sample mode and has an array to enable/disable measurement on each segment. Example test shows a dual sweep staircase IV curve.
PMU-1Ch-Sweep			This test performs a one-channel spot mean amplitude sweep.
PMU-1Ch-Wfm			This test performs a one-channel waveform capture.
PMU-SMU-IV-Sweep			This test to perform an IV sweep uses the output relays of the instrument cards to switch the SMU and PMU outputs to the device terminals.

Table 16-17 (continued)
Pulse project plans

Project plan/test	Project folder	Pulse generators supported	Description
PMU-Flash-NAND	**_Memory**	PMU, PGU	This project demonstrates the capabilities of the PMU for FLASH memory testing (see PMU-Flash-NAND project).
Program			This test uses a positive-going pulse to program the floating gate of a transistor.
Erase			This test uses a negative-going pulse to erase the charge from the floating gate of a transistor.
Fast-Program-Erase			This test generates sequence of program/erase pulses.
Vt-MaxGm			This test performs a threshold voltage measurement using the maximum GM method.
Vt-Meas			This test performs a threshold voltage measurement before programming.
Program-Endurance			This test uses a positive-going pulse to program the floating gate of a transistor.
Vt-Program			This test performs a threshold voltage measurement after programming.
Erase-Endurance			This test uses a negative-going pulse to erase the charge from the floating gate of a transistor.
Vt-Erase			This test performs a threshold voltage measurement after the erase pulse.
NVM_Examples	**_Memory**	PMU	This project has examples for characterization of NAND Flash, phase change memory (PCRAM), and ferroelectric memory (FeRAM). This project requires two SMUs, one 4225-PMU and two 4225-RPMs. See the NVM Application Note on the Applications page of the 4200 Complete Reference.
FLASH: Vt			This test uses the two SMUs to perform a Vt sweep on the flash test device.
FLASH: Program			This test applies a program pulse to the flash test device from the 4225-PMU.
FLASH: Vt_Programmed			This test uses the two SMUs to perform a Vt sweep on the flash test device to determine the programmed state voltage threshold.
FLASH: Erase			This test applies an erase pulse to the flash test device from the 4225-PMU.
FLASH: Vt_Erased			This test uses the two SMUs to perform a Vt sweep on the flash test device to determine the erased state voltage threshold
FLASH: Endurance			This stress/measure test applies an increasing number of program+erase waveforms to the flash test device from the 4225-PMU. Every log10 waveforms, a Vt measurement is performed with the SMUs (measure after 10, 100,1000, …).
PRAM: PRAM_Reset			This test applies a waveform consisting of a Reset pulse and a Reset resistance measure pulse from the 4225-PMU and 4225-RPM.

Table 16-17 (continued)
Pulse project plans

Project plan/test	Project folder	Pulse generators supported	Description
PRAM: PRAM_Set			This test applies a waveform consisting of a Set pulse and a Set resistance measure pulse from the 4225-PMU and 4225-RPM.
PRAM: IV_CURVE			This test applies a triangle pulse to the phase change test device and measures the I-V response.
PRAM: RI_CURVE			This test applies multiple Reset-measure-Set-measure waveforms to the test device. The Set pulse height is swept and measurements are extracted to create the RI curve.
PRAM: PRAM_Endurance			This stress/measure test applies an increasing number of Reset+Set waveforms to the phase-change test device from the 4225-PMU and 4225-RPM. Every log10 waveforms, the Reset and Set resistance measurements are taken (measure after 10, 100,1000, …).
FERAM: Hysteresis			This test applies two triangle voltage waveforms and plots the variation in the polarity charge versus the voltage.
FERAM: PUND			This test applies the four pulse waveform: Positive, Up, Negative, Down and extracts the measurements by calculating the charge during the pulse transitions.
FERAM: FERAM_Endurance			This stress/measure test applies an increasing number of PUND waveforms to the test device. The Psw and Qsw are measured and calculated every log10 waveforms (measure after 10, 100,1000, …).
PMU-MOSFET	**_Pulse**	PMU	This project includes tests to perform three-terminal, ultra fast IV and DC tests on a MOSFET. See PMU-MOSFET project for more information on this project.
DC-vds-id			This test uses two SMUs to generate a DC drain family of curves.
pulse-vds-id			This test uses CH1 and CH2 of a PMU to perform a pulsed drain family of curves.
SMU-PMU-vds-id			This test uses the output relays (not the RPMs) to switch the SMU and PMU outputs to the device terminals.
DC-vgs-id			This test measures DC vgs-id using SMUs.
pulse-vgs-id			This test measures pulsed IV vgs-id using PMUs.
scope-shot			This test performs waveform capture with spot means on the drain of a MOSFET.

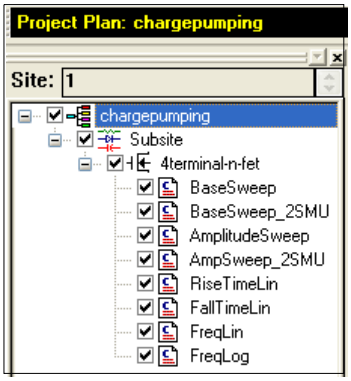
Table 16-17 (continued)
Pulse project plans

Project plan/test	Project folder	Pulse generators supported	Description
PMU-Switch	**_Pulse**	PMU	This project provides examples for switching between a PMU, SMU, and CVU to the device under test (DUT). See PMU-Switch project for details on using this project.
DC-IVsweep			This MOSFET test uses two SMUs to perform a DC drain family of curves.
pulse-IVsweep			This MOSFET test uses CH 1 and CH 2 of a PMU to generate a pulsed drain family of curves.
DC-IV-sweep			This diode test performs a DC I-V sweep of a diode using SMUs.
CV-sweep			Performs a C-V sweep on a diode.
pulse-IV-sweep			Performs a pulse I-V sweep on a diode.
SolarCell pulse_iv_sweep	**_Solar**	PMU	This test for the SolarCell project uses the PMU to perform a pulse IV sweep on the solar cell.

chargepumping project

The chargepumping project plan is shown in Figure 16-91.

Figure 16-91
chargepumping project plan



Key concepts

Charge pumping (CP) is a useful technique for understanding gate stack behavior, which is increasingly important as high κ films become more commonly used for transistor gates. CP characterizes interface and charge-trapping phenomena. The change in the CP results can be used to determine the amount of degradation caused by typical reliability test methods, employing either DC or pulsed stress.

Project summary

This project has eight user test modules (UTMs) for charge pumping testing:

- BaseSweep: The base voltage of the waveform is swept while the amplitude of the pulse if kept constant. The resulting charge pumping (I_{CP}) is measured as a function of the base voltage.
- BaseSweep_2SMU: Same as the BaseSweep UTM, except it uses a second SMU to apply a DC bias voltage to the source/drain terminals.
- AmplitudeSweep: The amplitude of the pulse is swept while the base voltage is kept constant. The charge pumping current (I_{CP}) is measured as a function of the pulse amplitude voltage.
- AmpSweep_2SMU: Same as the AmplitudeSweep user module, except that it uses a second SMU to apply a DC bias voltage to the source/drain terminals.
- RiseTimeLin: Performs a linear sweep of the rising transition time of the pulse. Charge pumping (I_{CP}) is measured as a function of the rise time.
- FallTimeLin: Performs a linear sweep of the falling transition time of the pulse. Charge pumping current (I_{CP}) is measured and graphed as a function of the fall time.
- FreqLin: With the amplitude, offset voltage, and rise/fall time kept constant, the (I_{CP}) is measured as a function of a linear sweep of the test frequency.
- FreqLog: With the amplitude, offset voltage, and rise/fall time kept constant, the charge pumping current (I_{CP}) is measured as a function of a log frequency sweep. It is a sweep of the test frequencies based on a multiplier-factor.

Opening the project plan

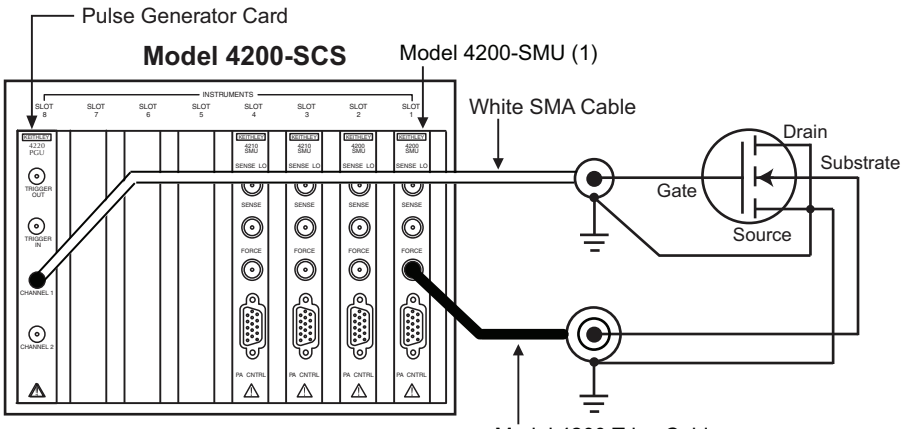
This project plan is opened as follows:

1. Click **File** at the top of the Keithley Interactive Test Environment (KITE) interface and select **Open Project** from the drop-down menu.
2. In the Open KITE Project File window, navigate back to the **Projects** folder.
3. Double-click the **_Pulse** folder.
4. Double-click the **chargepumping** folder.
5. Double-click **ChargePumping.kpr** to open the project.

Connections

For tests that require a single SMU, connect the SMU and pulse generator to the device under test (DUT) as shown in Figure 16-92. The source and drain of the DUT are connected to ground, while the gate is pulsed with a fixed frequency and amplitude. The substrate is connected to an SMU that is used to measure the current through the gate (I_{CP}).

Figure 16-92
Charge pumping—connections using one SMU



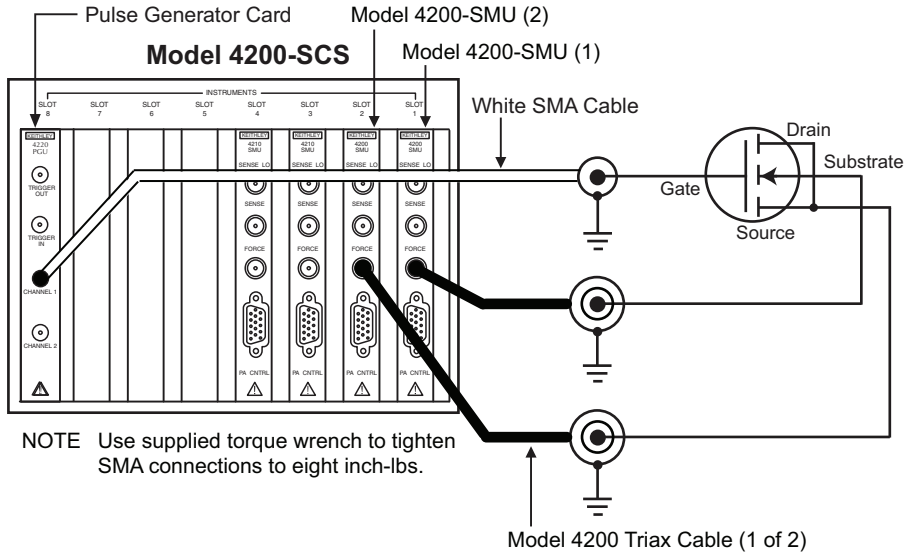
NOTE Use supplied torque wrench to tighten SMA connections to eight inch-lbs.

Use the single SMU connection scheme for the following UTMs:

- BaseSweep
- AmplitudeSweep
- RiseTimeLin
- FallTimeLin
- FreqLin
- FreqLog

For tests that require two SMUs and a pulse generator, connect the SMUs and pulse generator to the device under test (DUT) as shown in Figure 16-93. One SMU measures I_{CP}, and the other SMU applies a DC bias voltage to the source and drain terminals. The gate is pulsed with a fixed frequency and amplitude.

Figure 16-93
Charge pumping—connections using two SMUs



NOTE Use supplied torque wrench to tighten SMA connections to eight inch-lbs.

Use the two-SMU connection scheme for the following UTMs:

- BaseSweep_2SMU
- AmpSweep_2SMU

Running project plan tests

Running project plan tests provides the basic procedure to run the following project plan tests. Any special requirements will be explained in the documentation for the tests.

Renaming and running the project

Renaming the project

User test modules (UTMs) in the project plan are preconfigured using default settings. The tests have been run at the factory with generated test data and graphs saved in the project plan. The test data and graphs provide an example of expected results. If you want to retain the example test data and graphs, you can save this project under a different name.

With the project open, click **File** at the top of the KITE interface and select **Save Project As**. In the dialog box, make sure **Include data** is selected (checked). Type a new project name, select a location, and then click **OK**.

Running tests

Tests in the project are ready to run using the default settings. Running project plan tests provides the basic procedure to run the project plan tests. If you want to change the parameter settings for a test before running it, refer to the documentation for the Project plan tests.

Project plan tests

The following information provides details on these user test modules (UTMs) for the chargepumping project:

- BaseSweep
- BaseSweep_2SMU
- AmplitudeSweep
- AmpSweep_2SMU
- RiseTimeLin
- FallTimeLin
- FreqLin
- FreqLog

NOTE *The chargepumping project plan UTMs were created using the user modules that make up the chargepumping user library. For details, see the chargepumping user library. The documentation for the user library includes information on creating a new project plan (see the Procedure to perform charge pumping measurements).*

BaseSweep

The BaseSweep UTM was created using the BaseSweep user module of the chargepumping user library.

Test summary

The base voltage of the waveform is swept by a SMU, while the amplitude of the pulse generator output is kept constant. The resulting charge pumping current is measured as a function of base voltage. This routine graphs charge pumping current (I_{CP}) versus base voltage.

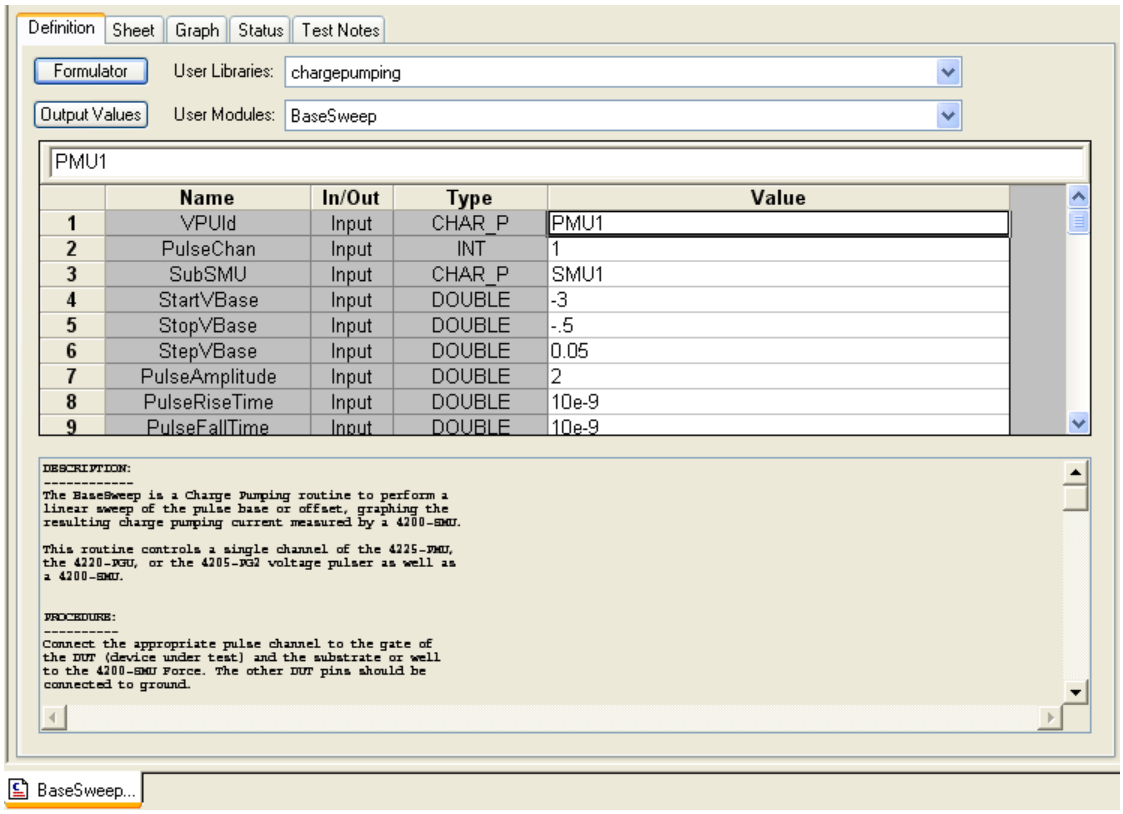
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click BaseSweep in the project navigator (see Figure 16-91) to open the test. Figure 16-94 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for BaseSweep for details on the input parameters.

Figure 16-94
Definition tab for BaseSweep



Inputs for BaseSweep

The input parameters are listed in Table 16-23. Input parameter values are set in the Definition tab for the BaseSweep test (see Figure 16-94). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for BaseSweep

The output parameters are listed in Table 16-24. When the test is run, test data is placed in the Data sheet for BaseSweep.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for BaseSweep

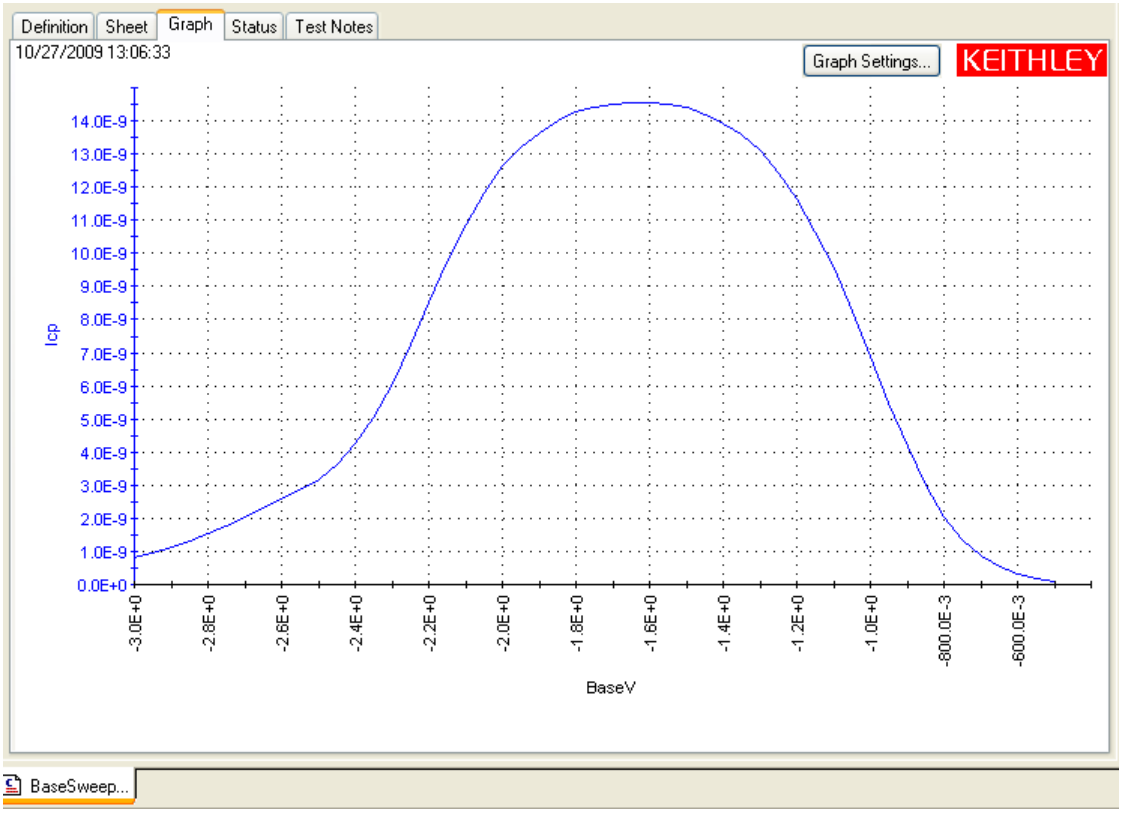
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- BaseV Base voltage values for the SMU voltage sweep.
- Icp Measured charge pumping current at each base voltage value.
- Qcp Calculated charge at each base voltage value (Qcp = Icp / Frequency).

Graph for BaseSweep

The graph is displayed in the Graph tab (see Figure 16-95).

Figure 16-95
Graph tab for BaseSweep



BaseSweep_2SMU

The BaseSweep_2SMU UTM was created using the BaseSweep_2SMU user module of the chargepumping user library.

Test summary

The base voltage of the waveform is swept by a SMU, while the amplitude of the pulse generator output is kept constant. The second SMU applies a DC voltage bias to the source and drain. The resulting charge pumping current is measured as a function of base voltage. This routine graphs charge pumping current (I_{CP}) versus base voltage.

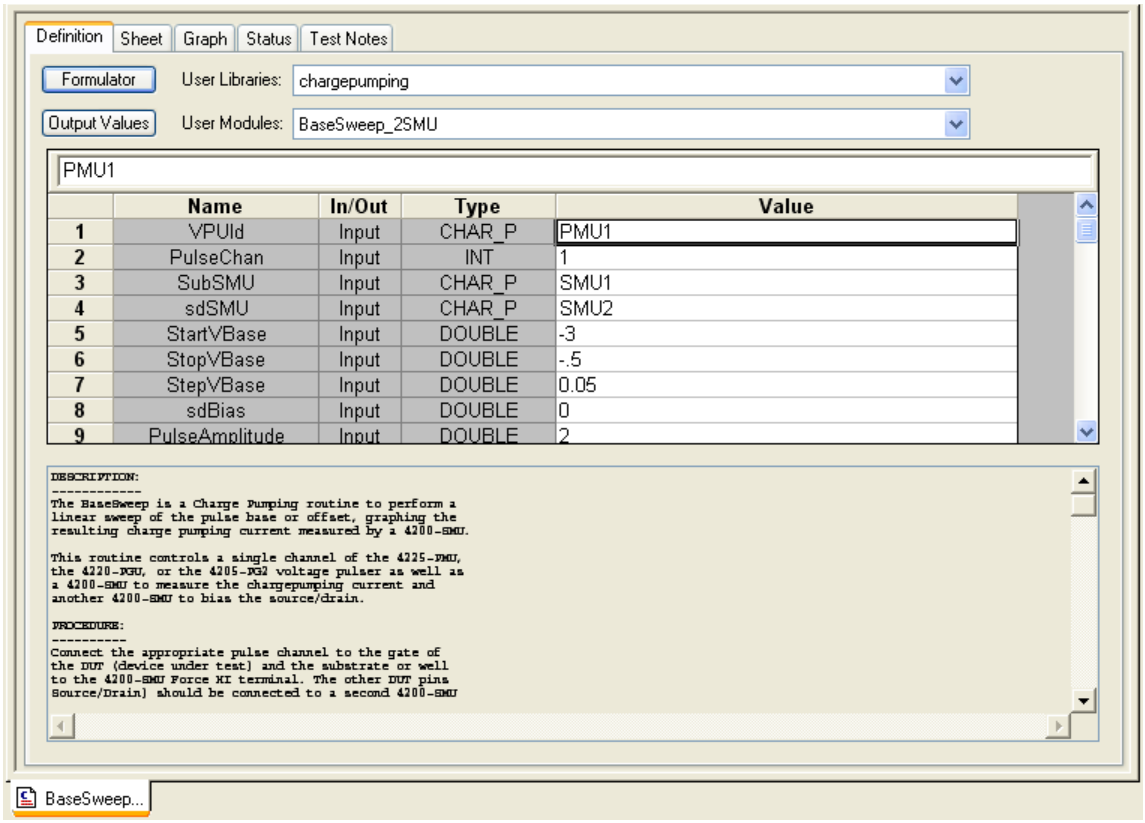
Connections

As shown in Figure 16-93, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other SMU to the source and drain.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click BaseSweep_2SMU in the project navigator (see Figure 16-91) to open the test. Figure 16-96 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for BaseSweep_2SMU for details on the input parameters.

Figure 16-96
Definition tab for BaseSweep_2SMU



Inputs for BaseSweep_2SMU

The input parameters are listed in Table 16-25. Input parameter values are set in the Definition tab for the BaseSweep_2SMU test (see Figure 16-96). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 to 90 percent times. For the 5 V range of the pulse generator, the 10 to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for BaseSweep_2SMU

The output parameters are listed in Table 16-26. When the test is run, test data is placed in the Data sheet for BaseSweep_2SMU.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for BaseSweep_2SMU

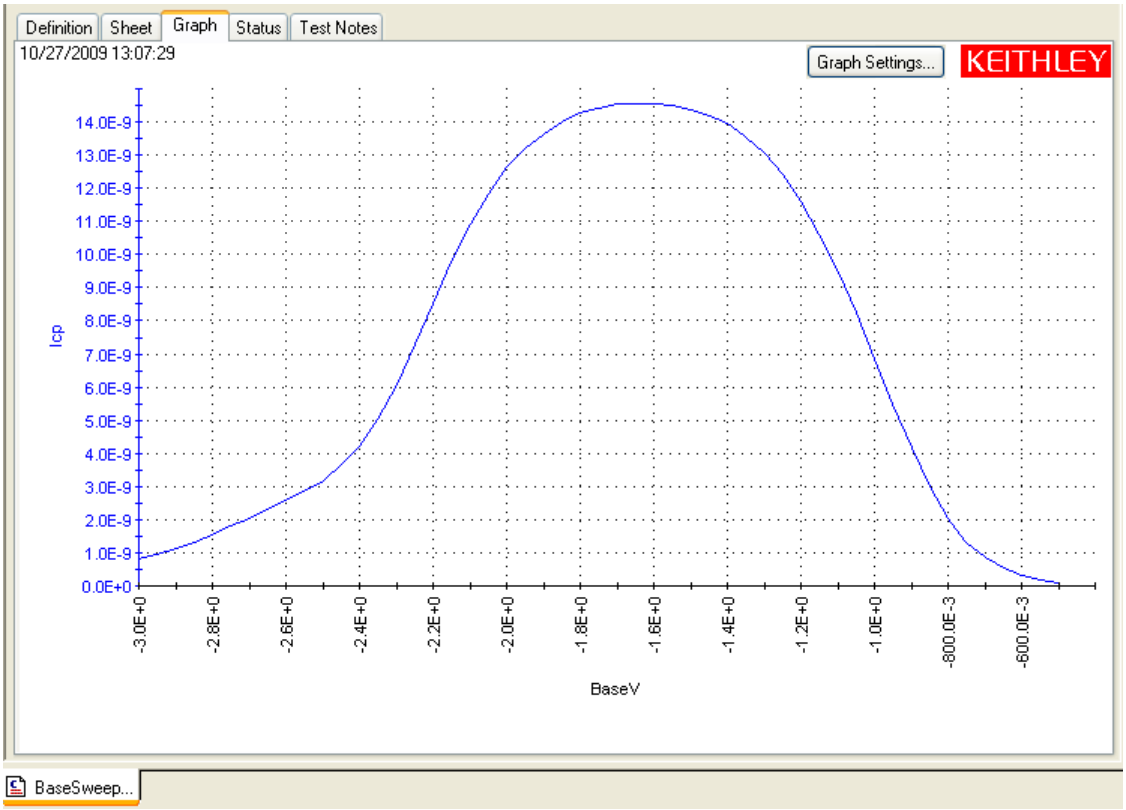
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- BaseV Base voltage values for the SMU voltage sweep.
- Icp Measured charge pumping current at each base voltage value.
- Qcp Calculated charge at each base voltage value (Qcp = Icp / Frequency).

Graph for BaseSweep_2SMU

The graph is displayed in the Graph tab (see Figure 16-97).

Figure 16-97
Graph tab for BaseSweep_2SMU



AmplitudeSweep

The AmplitudeSweep UTM was created using the AmplitudeSweep user module of the chargepumping user library.

Test summary

The amplitude of the pulse generator output is swept, while the SMU base voltage is kept constant. The charge pumping current is measured as a function of pulse amplitude voltage. This routine graphs charge pumping current (I_{CP}) versus pulse amplitude.

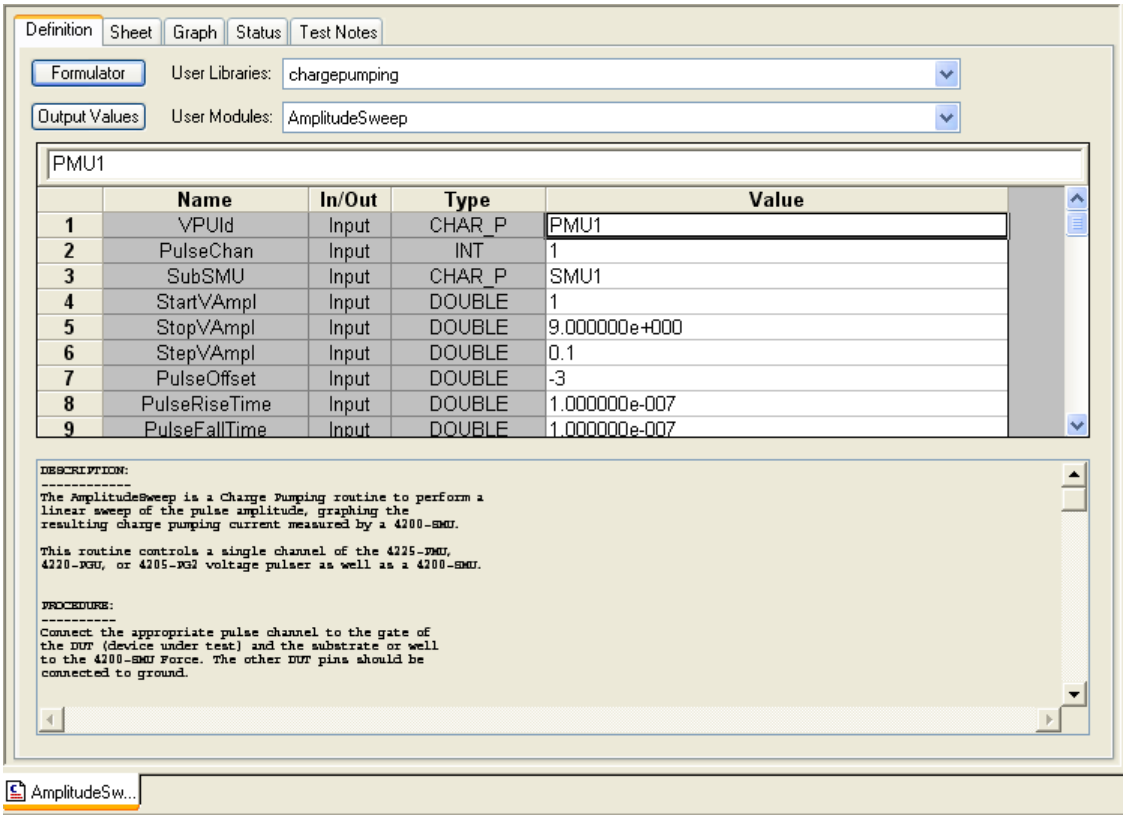
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click AmplitudeSweep in the project navigator (see Figure 16-91) to open the test. Figure 16-98 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for AmplitudeSweep for details on the input parameters.

Figure 16-98
Definition tab for AmplitudeSweep



Inputs for AmplitudeSweep

The input parameters are listed in Table 16-19. Input parameter values are set in the Definition tab for the AmplitudeSweep test (see Figure 16-98). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for AmplitudeSweep

The output parameters are listed in Table 16-20. When the test is run, test data is placed in the Data sheet for AmplitudeSweep.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for AmplitudeSweep

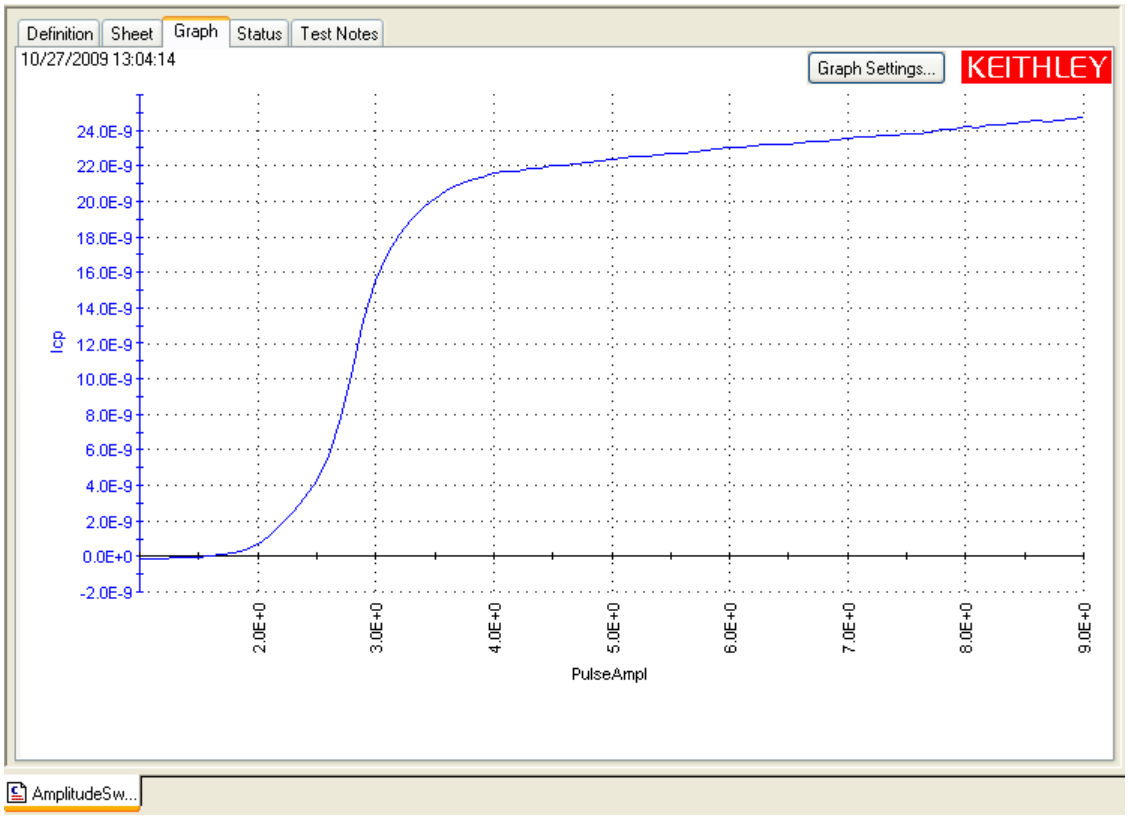
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- PulseAmpl Pulse amplitude values for the pulse voltage sweep.
- Icp Measured charge pumping current at each pulse amplitude value.
- Qcp Calculated charge at each pulse voltage value (Qcp = Icp / Frequency).

Graph for AmplitudeSweep

The graph is displayed in the Graph tab (see Figure 16-99).

Figure 16-99
Graph tab for AmplitudeSweep



AmpSweep_2SMU

The AmpSweep_2SMU UTM was created using the AmplitudeSweep_2SMU user module of the chargepumping user library.

Test summary

The amplitude of the pulse generator output is swept, while the SMU base voltage is kept constant. The second SMU applies a DC voltage bias to the source and drain. The charge pumping current is measured as a function of pulse amplitude voltage. This routine graphs charge pumping current (I_{CP}) versus pulse amplitude.

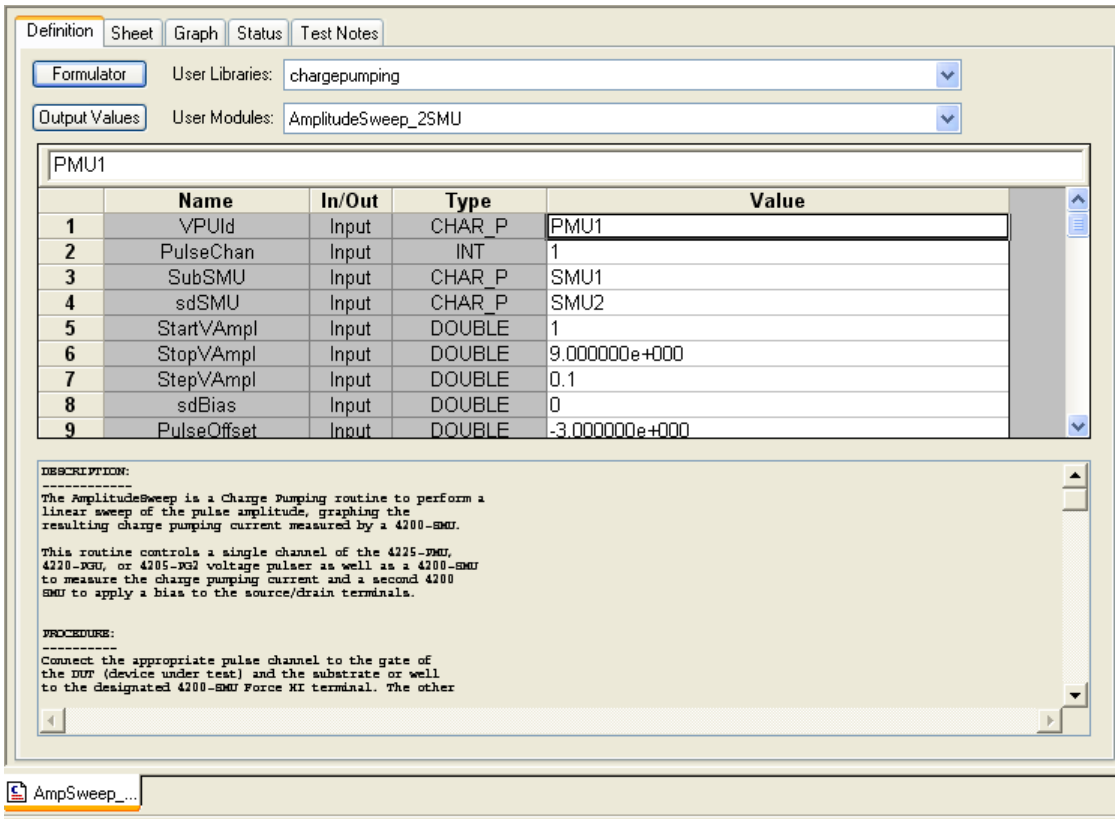
Connections

As shown in Figure 16-93, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click **Amplitude_2SMU** in the project navigator (see Figure 16-91) to open the test. Figure 16-100 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for AmpSweep_2SMU for details on the input parameters.

Figure 16-100
Definition tab for AmpSweep_2SMU



Inputs for AmpSweep_2SMU

The input parameters are listed in Table 16-21. Input parameter values are set in the Definition tab for the AmpSweep_2SMU test (see Figure 16-100). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for AmpSweep_2SMU

The output parameters are listed in Table 16-22. When the test is run, test data is placed in the Data sheet for AmpSweep_2SMU.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for AmpSweep_2SMU

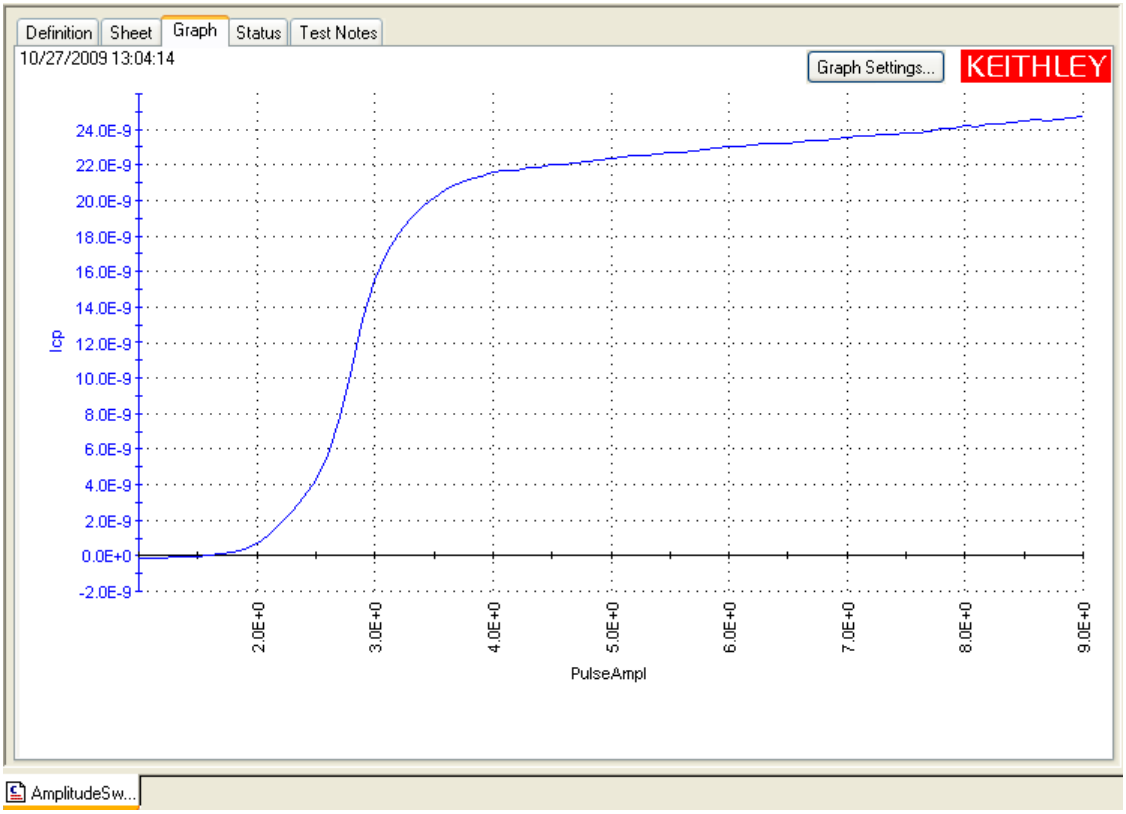
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- PulseAmpl Pulse amplitude values for the pulse voltage sweep.
- Icp Measured charge pumping current at each pulse amplitude value.
- Qcp Calculated charge at each pulse voltage value (Qcp = Icp / Frequency).

Graph for AmpSweep_2SMU

The graph is displayed in the Graph tab (see Figure 16-101).

Figure 16-101
Graph tab for AmpSweep_2SMU



RiseTimeLin

The RiseTimeLin UTM was created using the RiseTimeLinearSweep user module of the chargepumping user library.

Test summary

The rise time of the pulse generator output is swept, while the SMU base voltage is kept constant. The charge pumping current is measured as a function of pulse rise time. This routine graphs charge pumping current (I_{CP}) versus pulse rise time.

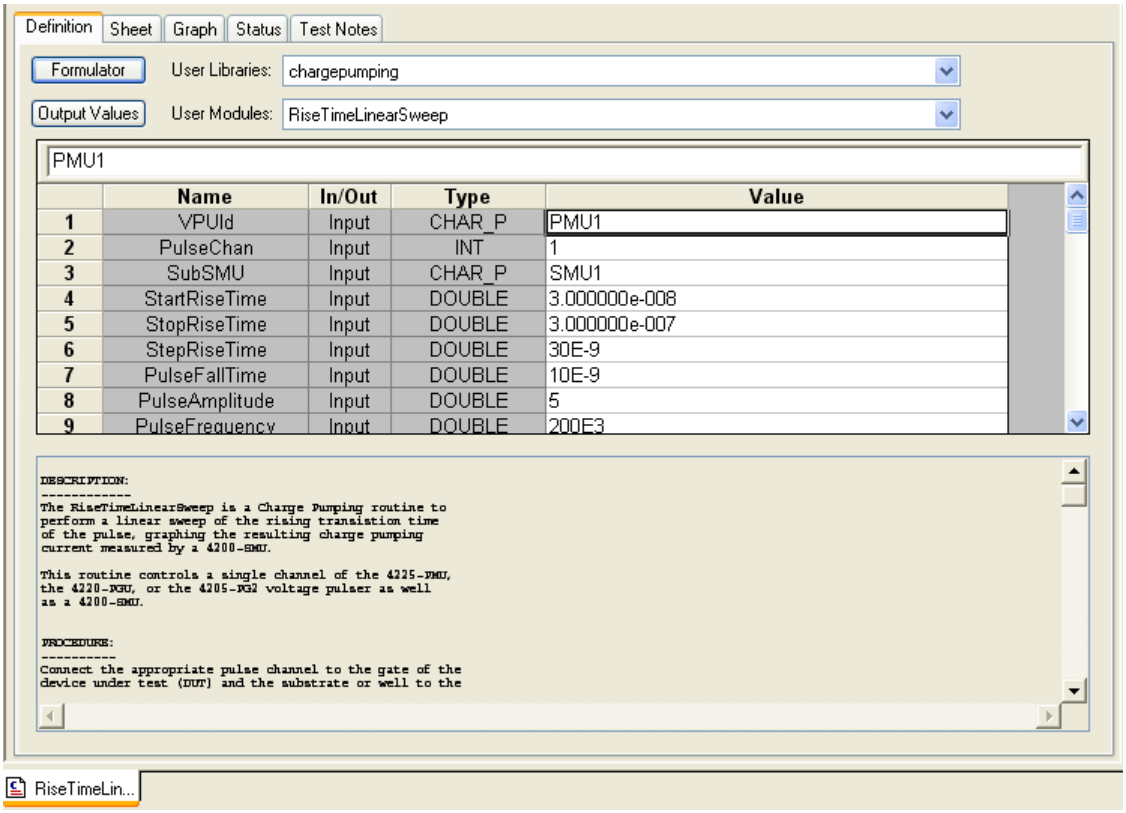
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click **RiseTimeLin** in the project navigator (see Figure 16-91) to open the test. Figure 16-102 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for RiseTimeLin for details on the input parameters.

Figure 16-102
Definition tab for RiseTimeLin



Inputs for RiseTimeLin

The input parameters are listed in Table 16-33. Input parameter values are set in the Definition tab for the RiseTimeLin test (see Figure 16-102). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for RiseTimeLin

The output parameters are listed in Table 16-34. When the test is run, test data is placed in the Data sheet for RiseTimeLin.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for RiseTimeLin

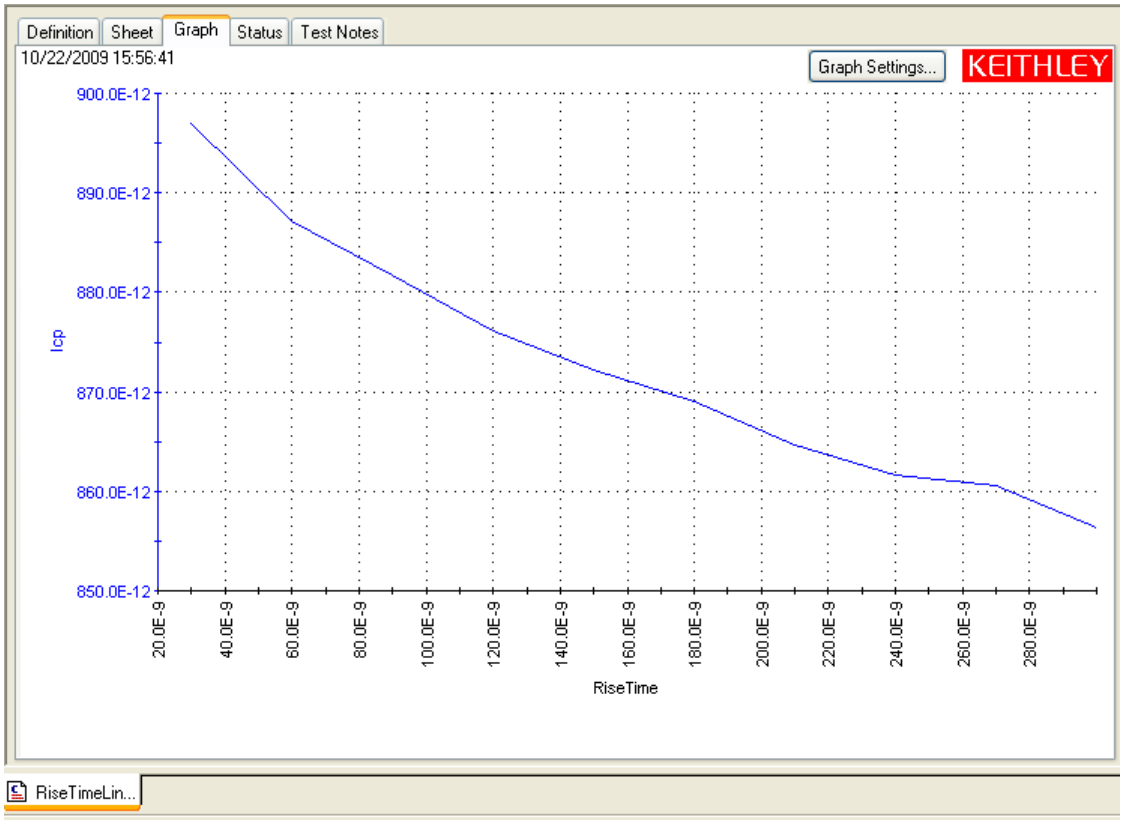
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- RiseTime Rise time values for the pulse rise time sweep.
- Icp Measured charge pumping current at each pulse rise time value.
- Qcp Calculated charge at each pulse rise time value (Qcp = Icp / Frequency).

Graph for RiseTimeLin

The graph is displayed in the Graph tab (see Figure 16-103).

Figure 16-103
Graph tab for RiseTimeLin



FallTimeLin

The FallTimeLin UTM was created using the FallTimeLinearSweep user module of the chargepumping user library.

Test summary

The fall time of the pulse generator output is swept, while the SMU base voltage is kept constant. The charge pumping current is measured as a function of pulse fall time. This routine graphs charge pumping current (I_{CP}) versus pulse fall time.

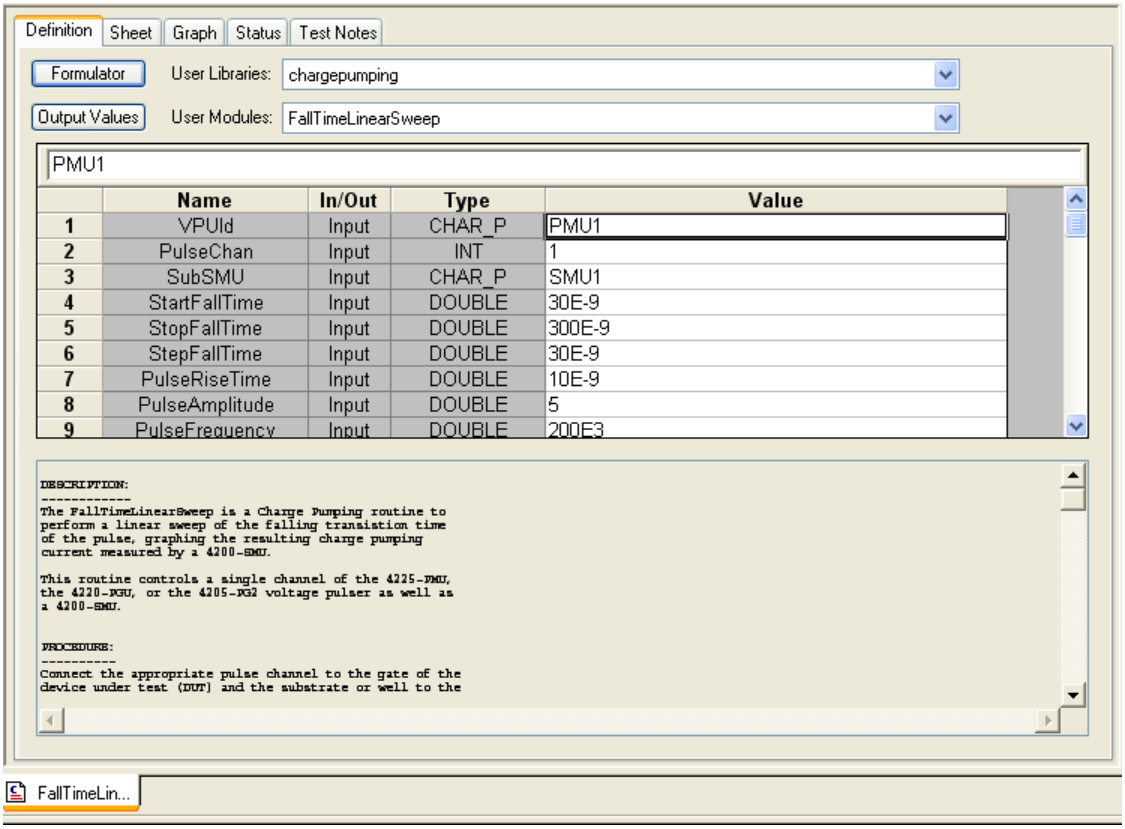
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click **FallTimeLin** in the project navigator (see Figure 16-91) to open the test. Figure 16-104 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for FallTimeLin for details on the input parameters.

Figure 16-104
Definition tab for FallTimeLin



Inputs for FallTimeLin

The input parameters are listed in Table 16-27. Input parameter values are set in the Definition tab for the FallTimeLin test (see Figure 16-102). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for FallTimeLin

The output parameters are listed in Table 16-28. When the test is run, test data is placed in the Data sheet for FallTimeLin.

Return values

When a test is run, status of the test is reported by return values (see Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for FallTimeLin

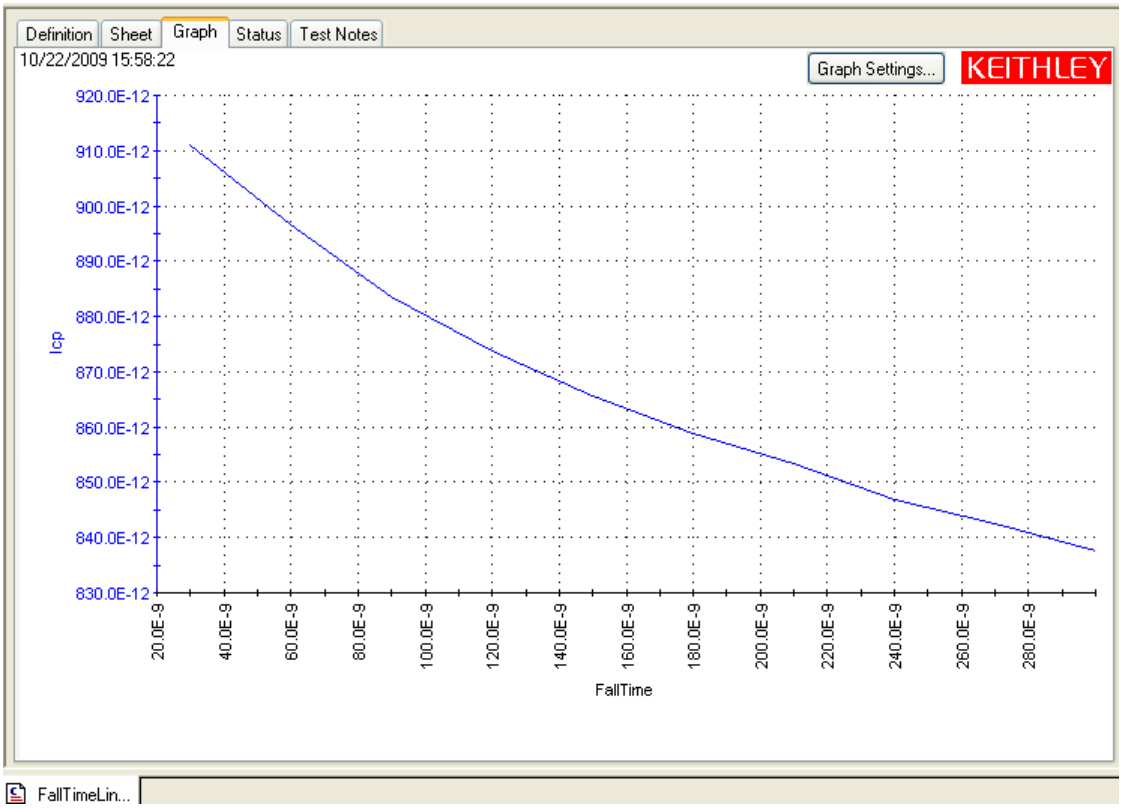
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- FallTime Fall time values for the pulse fall time sweep.
- Icp Measured charge pumping current at each pulse fall time value.
- Qcp Calculated charge at each pulse fall time value (Qcp = Icp / Frequency).

Graph for FallTimeLin

The graph is displayed in the Graph tab (see Figure 16-105).

Figure 16-105
Graph tab for FallTimeLin



FreqLin

The FreqLin UTM was created using the FreqLinearSweep user module of the chargepumping user library.

Test summary

A linear sweep of pulse frequency is performed, while the SMU base voltage is kept constant. The charge pumping current is measured as a function of frequency. This routine graphs charge pumping current (I_{CP}) versus frequency.

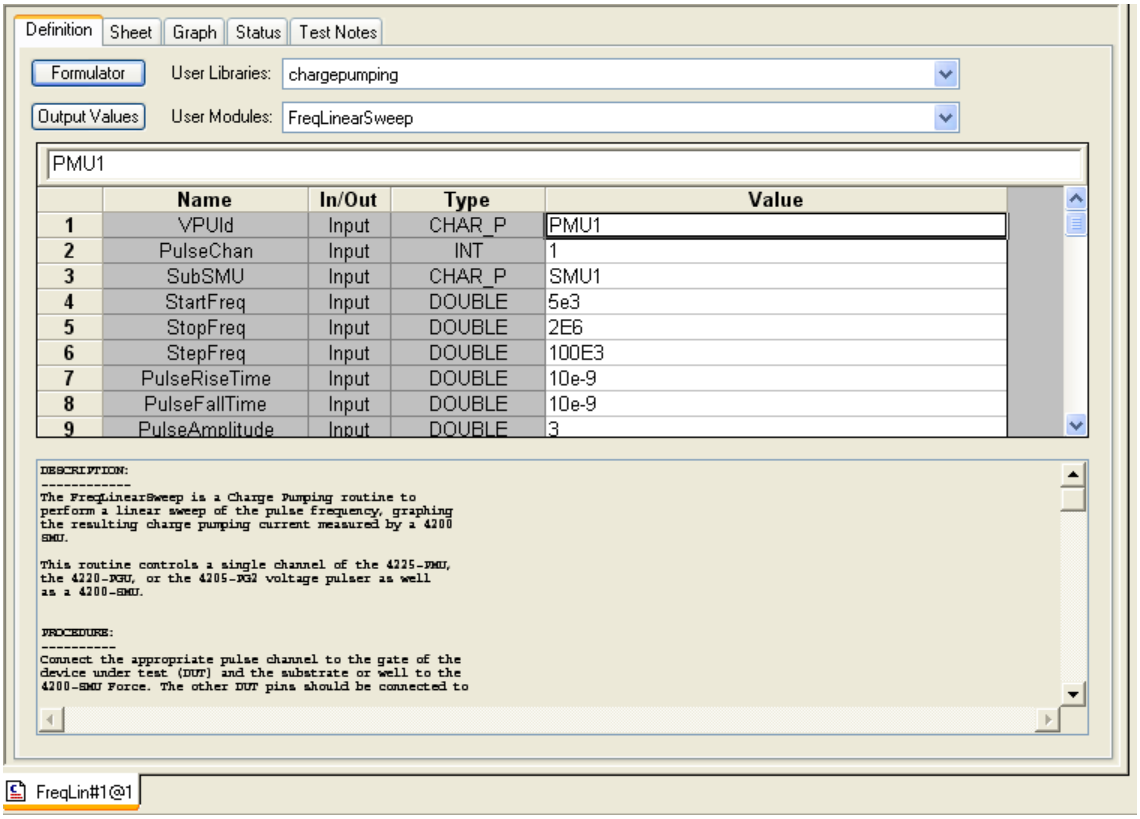
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click **FreqLin** in the project navigator (see Figure 16-91) to open the test. Figure 16-102 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for FreqLin for details on the input parameters.

Figure 16-106
Definition tab for FreqLin



Inputs for FreqLin

The input parameters are listed in Table 16-31. Input parameter values are set in the Definition tab for the FallTimeLin test (see Figure 16-106). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

Outputs for FreqLin

The output parameters are listed in Table 16-32. When the test is run, test data is placed in the Data sheet for FreqLin.

Return values

When a test is run, status of the test is reported by return values (See Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for FreqLin

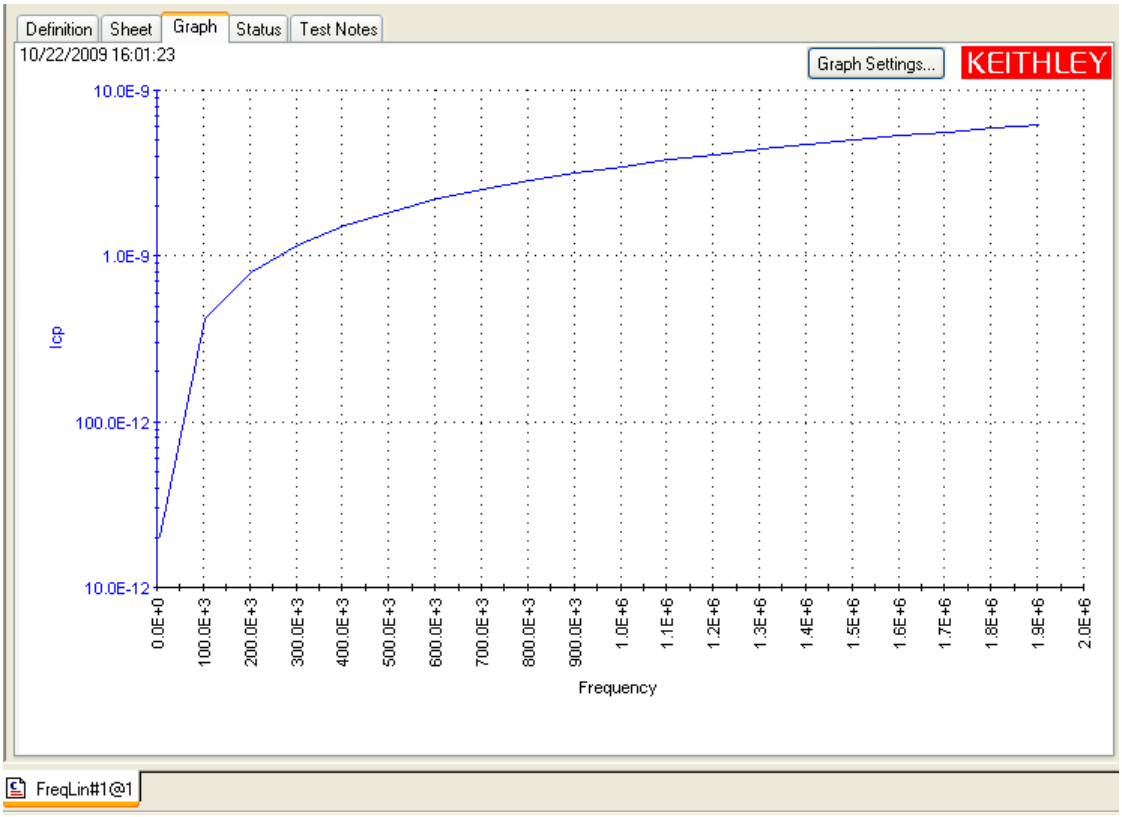
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- Frequency Frequency values for the pulse frequency sweep.
- Icp Measured charge pumping current at each pulse frequency value.
- Qcp Calculated charge at each pulse frequency value (Qcp = Icp / Frequency).

Graph for FreqLin

The graph is displayed in the Graph tab (see Figure 16-105).

Figure 16-107
Graph tab for FreqLin



FreqLog

The FreqLog UTM was created using the FreqFactorSweep user module of the chargepumping user library.

Test summary

A logarithmic (or multiplier-factor) sweep of pulse frequency is performed, while the SMU base voltage is kept constant. The charge pumping current is measured as a function of frequency. This routine graphs charge pumping current (I_{CP}) versus frequency.

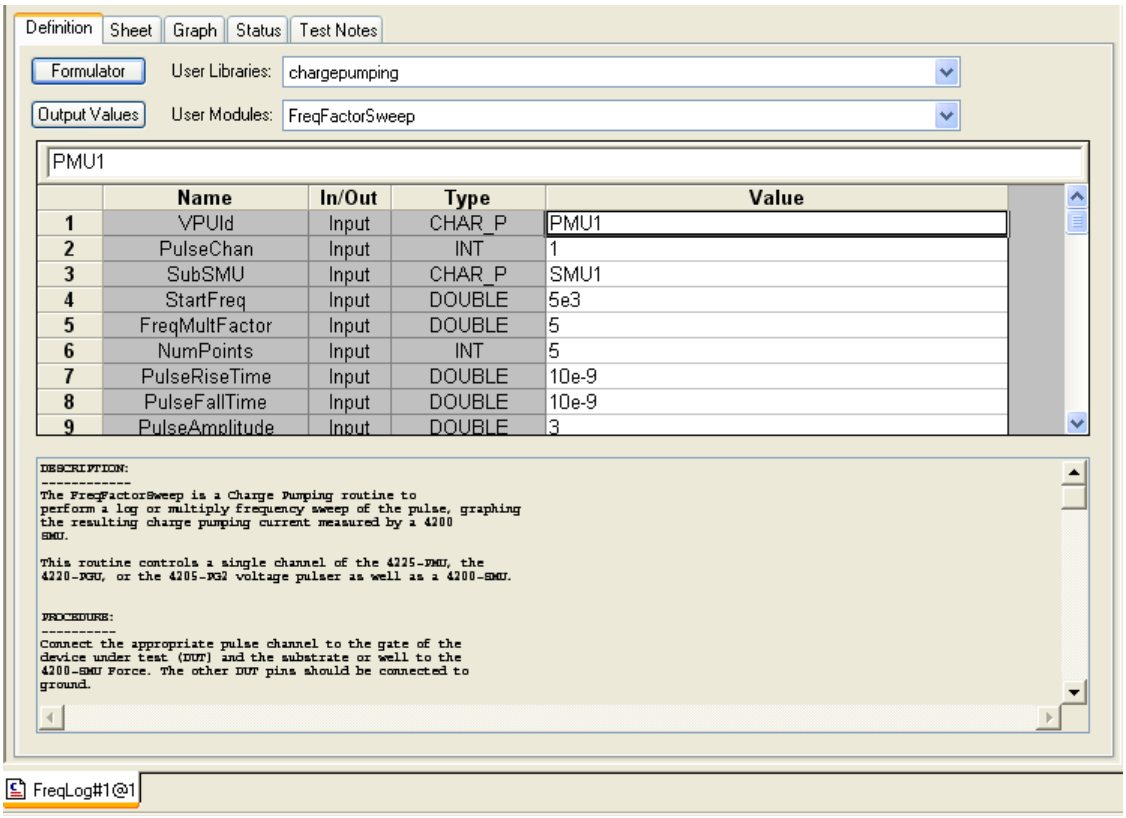
Connections

As shown in Figure 16-92, connect the appropriate pulse generator channel to the gate of the DUT (device under test) and the SMU Force terminal to the substrate (or well). Connect the other DUT pins to ground.

Definition tab

With the chargepumping project plan open (see Opening the project plan), double-click **FreqLog** in the project navigator (see Figure 16-91) to open the test. Figure 16-108 shows the Definition tab for the test. Use the **Definition** tab to enter the input parameters for the test. The parameter values shown in the Definition tab are the default settings. See Inputs for FreqLog for details on the input parameters.

Figure 16-108
Definition tab for FreqLog



Inputs for FreqLog

The input parameters are listed in Table 16-29. Input parameter values are set in the Definition tab for the FallTimeLin test (see Figure 16-108). The rise time and fall time parameters are the full transition times (zero to100 percent), not the 10 percent to 90 percent times. For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to100 percent transition times that are used to program the pulse.

The StartFreq, FreqMultFactor, and NumPoints input parameters are used to set the frequency values for the sweep. The FreqMultFactor value is the multiplier applied to the start frequency (StartFreq) for the sweep. The NumPoints value specifies the number frequency points in the sweep. Example:

StartFreq = 5E3 (5kHz)
FreqMultFactor = 5
NumPoints = 4

Freq1 = StartFreq
= 5 kHz

Freq2 = Freq1 x FreqMultFactor
= 5 kHz x 5
= 25 kHz

Freq3 = Freq2 x FreqMultFactor
= 25 kHz x 5
= 125 kHz

Freq4 = Freq3 x FreqMultFactor
= 125 kHz x 5
= 625 kHz

Outputs for FreqLog

The output parameters are listed in Table 16-30. When the test is run, test data is placed in the Data sheet for FreqLog.

Return values

When a test is run, status of the test is reported by return values (See Return values for details). A return value of zero indicates that the test ran properly. Returned negative numbers are errors.

Data sheet for FreqLog

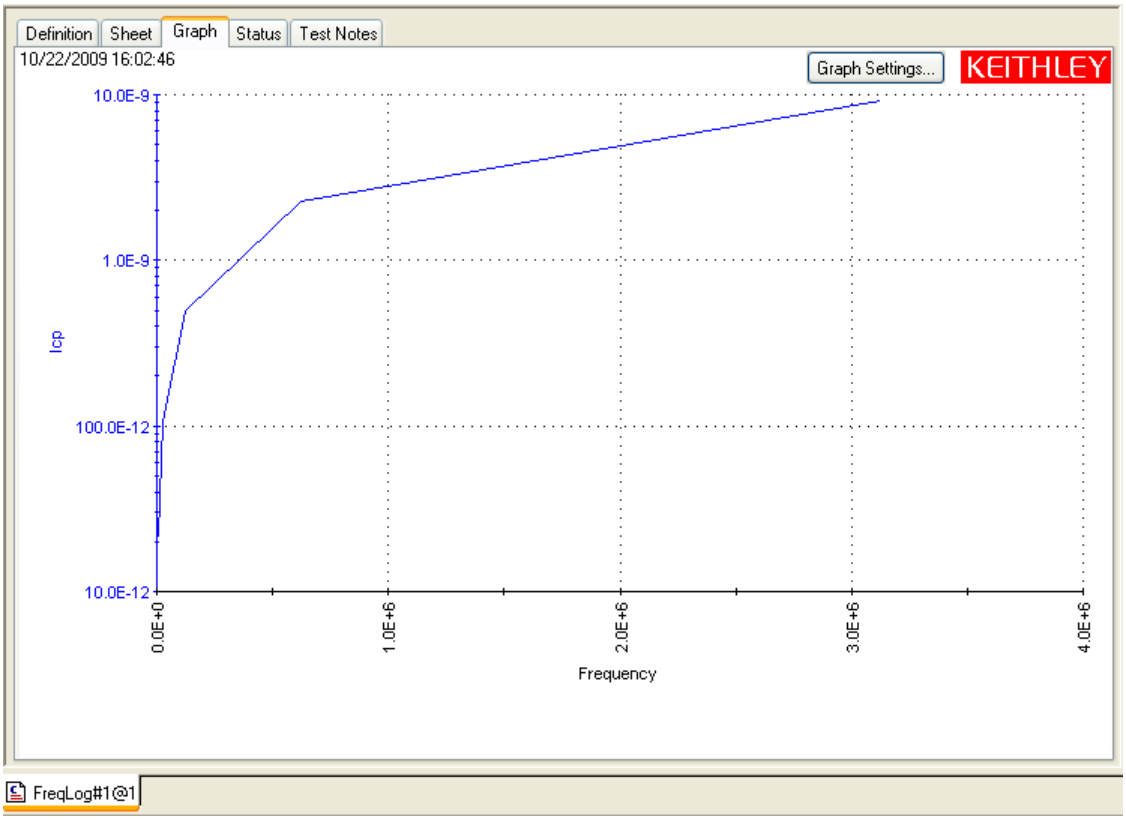
Test data is displayed in the Sheet tab (example shown in Figure 16-111) using the following array names:

- Frequency Frequency values for the pulse frequency sweep.
- Icp Measured charge pumping current at each pulse frequency value.
- Qcp Calculated charge at each pulse frequency value (Qcp = Icp / Frequency).

Graph for FreqLog

The graph is displayed in the Graph tab (see Figure 16-109).

Figure 16-109
Graph tab for FreqLog



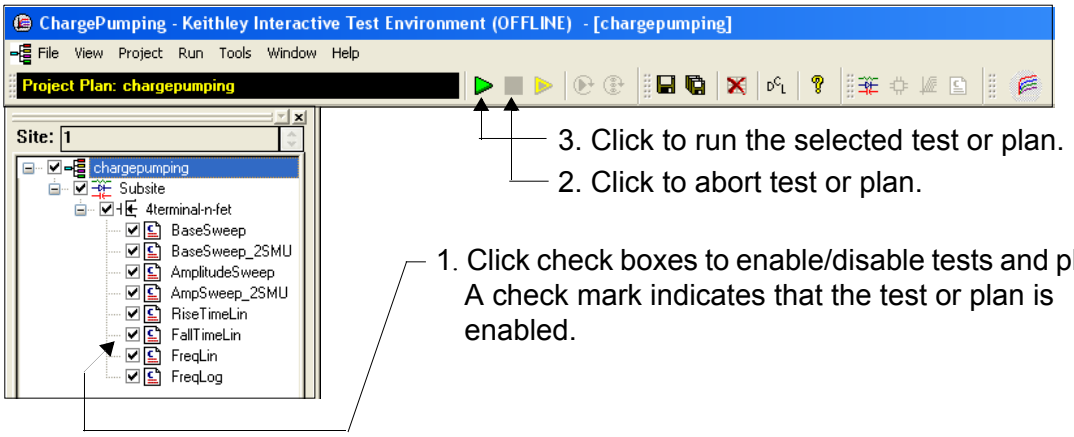
Running project plan tests

A single UTM (user test module) can be run, or multiple tests can be executed by running a device plan or project plan. Only an enabled test or plan can be run.

Perform the following steps to run an individual test or all the tests in the project:

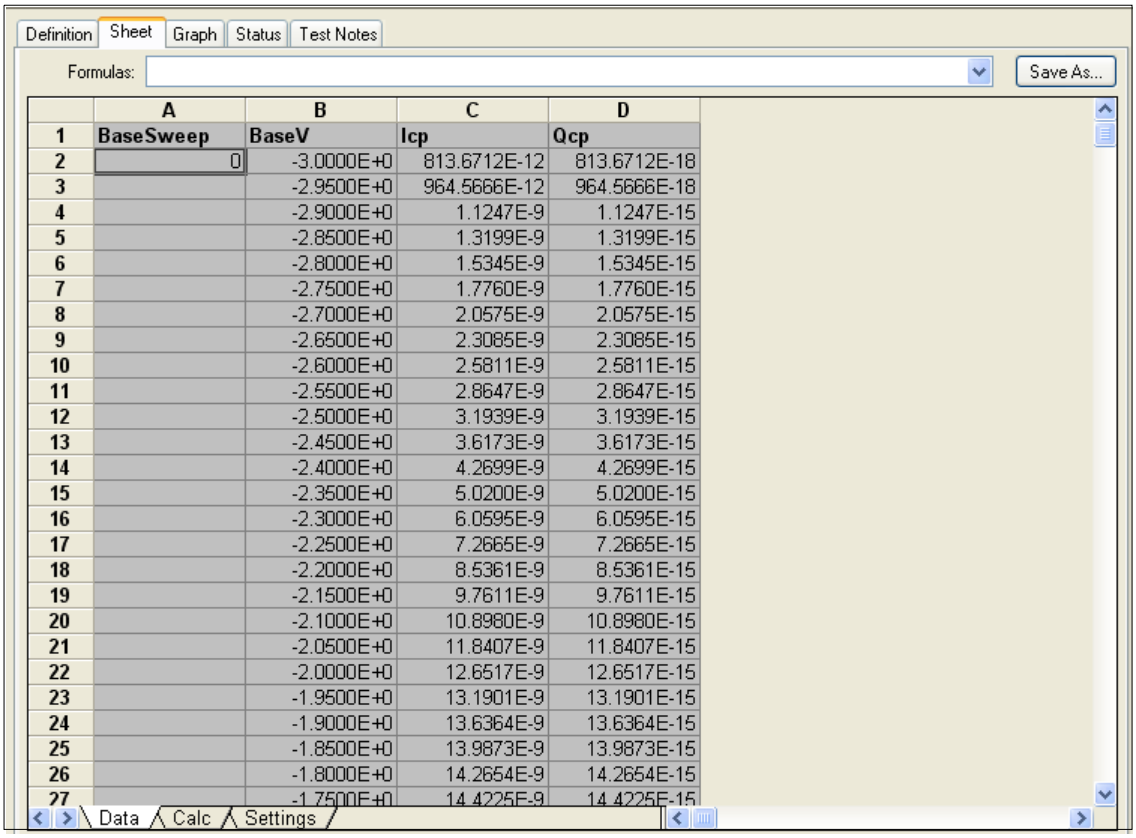
1. Connect the pulse generator and the SMU (or SMUs) to the device under test (DUT) as shown in Figure 16-92 or Figure 16-93.
2. Configure the tests, as explained in the documentation for the UTMs.
3. Perform the steps in Figure 16-110 to run a single test or all the tests in the project plan. The project plan navigator shown in Figure 16-110 shows all tests and plans enabled (checked). To run the first UTM only, select the **BaseSweep** UTM, and then run it. To run the entire project plan (all tests), select the **4terminal-n-fet** project plan, and then run it.

Figure 16-110
Running a test or plan



4. Analyze the test data in the Sheet and Graph tabs. All the UTMs generate a graph. Figure 16-111 shows an example of a Sheet tab. The graph generated by a UTM is provided with the documentation for the UTM.

Figure 16-111
Sheet tab for test data



	A	B	C	D
1	BaseSweep	BaseV	Icp	Qcp
2	0	-3.0000E+0	813.6712E-12	813.6712E-18
3		-2.9500E+0	964.5666E-12	964.5666E-18
4		-2.9000E+0	1.1247E-9	1.1247E-15
5		-2.8500E+0	1.3199E-9	1.3199E-15
6		-2.8000E+0	1.5345E-9	1.5345E-15
7		-2.7500E+0	1.7760E-9	1.7760E-15
8		-2.7000E+0	2.0575E-9	2.0575E-15
9		-2.6500E+0	2.3085E-9	2.3085E-15
10		-2.6000E+0	2.5811E-9	2.5811E-15
11		-2.5500E+0	2.8647E-9	2.8647E-15
12		-2.5000E+0	3.1939E-9	3.1939E-15
13		-2.4500E+0	3.6173E-9	3.6173E-15
14		-2.4000E+0	4.2699E-9	4.2699E-15
15		-2.3500E+0	5.0200E-9	5.0200E-15
16		-2.3000E+0	6.0595E-9	6.0595E-15
17		-2.2500E+0	7.2665E-9	7.2665E-15
18		-2.2000E+0	8.5361E-9	8.5361E-15
19		-2.1500E+0	9.7611E-9	9.7611E-15
20		-2.1000E+0	10.8980E-9	10.8980E-15
21		-2.0500E+0	11.8407E-9	11.8407E-15
22		-2.0000E+0	12.6517E-9	12.6517E-15
23		-1.9500E+0	13.1901E-9	13.1901E-15
24		-1.9000E+0	13.6364E-9	13.6364E-15
25		-1.8500E+0	13.9873E-9	13.9873E-15
26		-1.8000E+0	14.2654E-9	14.2654E-15
27		-1.7500E+0	14.4225E-9	14.4225E-15

Return values

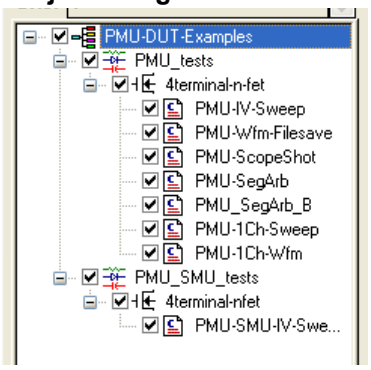
When a test is run, status of the test is reported as return values:

0	OK
-1	Invalid VPUId
-2	Invalid pulse channel selected
-3	Invalid SMUId
-4	Invalid range selected for the pulse generator
-5	Invalid start voltage for the base
-6	Invalid stop voltage for the base
-7	Invalid amplitude for the pulse generator at given range
-8	Invalid step voltage; the step is too large for given start and stop
-9	Invalid rise time for given pulse generator range
-10	Invalid fall time for given pulse generator range
-11	Invalid combination of rise time, fall time, frequency, and bandwidth
-12	Invalid load for the pulse generator
-13	Invalid NPLC for the SMU
-14	Invalid limited autorange for SMU
-15	Invalid compliance for SMU
-16	Array sizes do not match
-17	Arrays not large enough for the configured sweep
-18	Pulse offset invalid
Others	Negative numbers are errors. Refer to LPT and PulseIV documentation for details.

PMU-DUT-Examples project

The tests for this project are summarized in Table 16-17. Figure 16-112 shows the project navigator for this project. There are two main purposes of the tests in this project: 1) Provide test types that are not available in ITMs; 2) Provide simple examples for coding a PMU UTM.

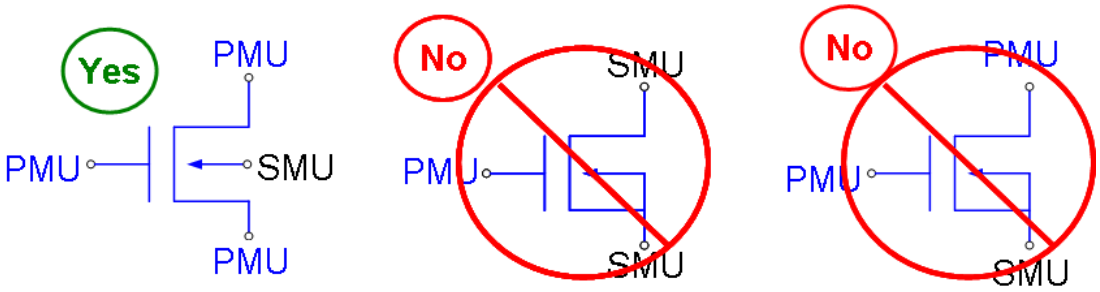
Figure 16-112
Project navigator for the PMU-DUT-Examples project



Using a SMU with PMUs

A SMU can only be used with PMUs if it is used to DC bias (no measure) an uncoupled DUT (device under test) pin. Uncoupled means that the pin does not respond to the PMU pulse. An example is the bulk terminal of a CMOS device. The gate can also be DC biased by an SMU when pulsing the drain or source. Figure 16-113 shows examples of valid and invalid connection schemes for PMUs and SMUs

Figure 16-113
Valid and invalid connection schemes



Test setups

Figure 16-114 shows the setup for the six PMU tests. These tests can also use RPMs (not shown in the illustration). The setup shows the substrate of the MOSFET connected to ground. This test also allows you to bias the substrate by breaking the ground connection and connecting the substrate to a SMU.

Figure 16-114
Test setup for the PMU tests

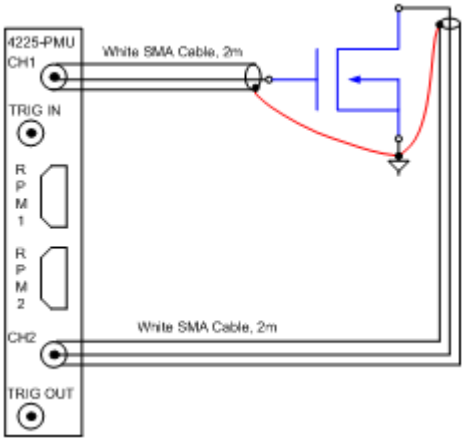
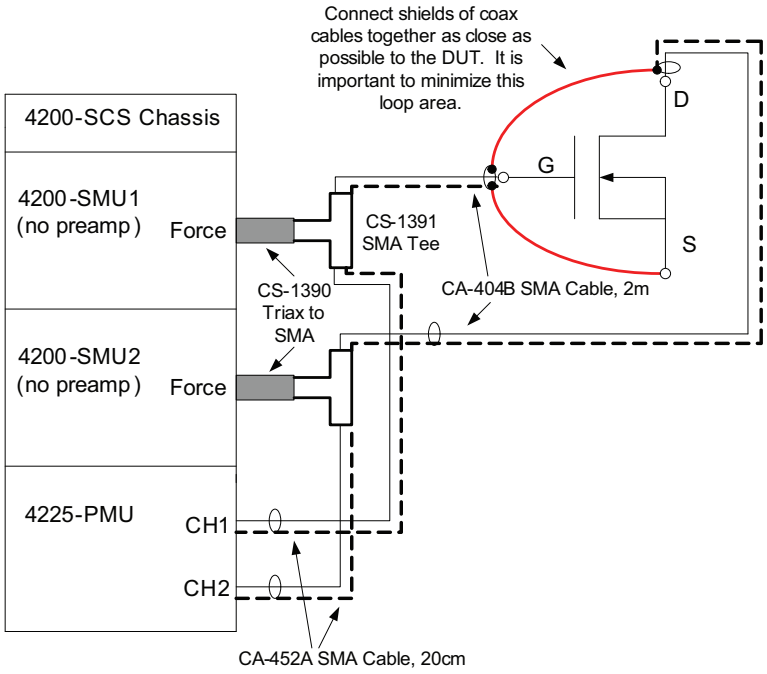


Figure 16-115 shows the test setup for the PMU-SMU-IV-Sweep test. This test performs switching between the SMU and PMU using the output relays on the instrument cards.

Figure 16-115
MOSFET test setup (uses output relays for switching)



Cabling from Two 4200-SMUs and 4225-PMU to Three-Terminal MOSFET:

Keithley P/N	QTY	Description
CA-404B	2	SMA to SMA Cable, 2m (supplied with PMU)
CS-1390	2	Male Triax to Female SMA Adaptor (guard removed)*
CS-1391	2	SMA Tee F-M-F*
CA-452A	2	SMA to SMA cable, 20cm*
N/A	2	Jumper Wires to Connect Shields (user supplied)

*included in Model 4200-PMU-PROBER-KIT

Do not connect the SMU to the drain, source, collector, or emitter of a transistor. If the DUT is not a transistor, do not connect the SMU to any pin that has a coupled or high-speed response to the pulses applied to other DUT terminals.

Test descriptions

User test modules (UTMs) are used for this project. See Section 6 in the User's Manual for an introduction to programming the Model 4200-SCS using KULT and Section 8 of this Reference Manual for advanced information and card programming command reference.

PMU-IV-Sweep

This test performs a two-channel pulse IV test on a MOSFET. One channel of the PMU applies a pulse train to the gate and the other channel of the same PMU performs an amplitude sweep for the drain.

In addition to optional SMU bias, the unique feature of this test is the ability to independently set the pulse delay, width, rise, and fall times for the two channels in the test. The unique timing parameters are PulseWidthGate, RiseTimeGate, FallTimeGate, DelayGate, PulseWidthDrain, RiseTimeDrain, FallTimeDrain, and DelayDrain. The pulse period is the same for both channels, therefore there is only one period parameter.

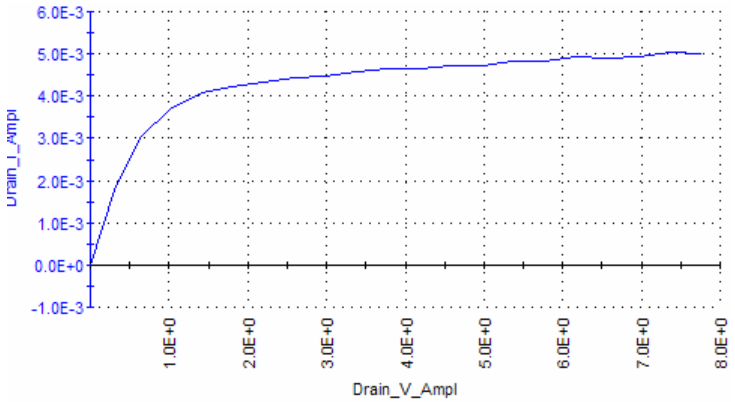
There is a restriction on pulse timing. The spot mean measure window, defined for the gate channel, must overlap the pulse top of the drain channel. The independent timing allows for dispersion characterization, which is typically performed on compound semiconductor (III-V) devices. For independent timing on a waveform capture, see the PMU-ScopeShot test.

This test supports load line effect compensation (LLEC), as well as current measure ranging and thresholds. To enable LLEC, set LoadLineGate and/or LoadLine Drain = 1. For limited autoranging, set IRangeGate or IRangeDrain to the desired current range, then set LtdCurrAutoGate or LtdCurrAutoDrain = 1. For thresholds, set to the desired value. Then set PMUMode = 1. To disable LLEC, ranging, and thresholds, use PMUMode = 0.

A SMU can be used to provide voltage bias for the substrate. Do not connect the SMU to a DUT drain, source, collector, or emitter since unexpected results could occur.

Figure 16-114 shows the setup for this test and Figure 16-116 shows an example graph.

Figure 16-116
Graph for PMU-IV-Sweep



PMU-Wfm-FileSave

This test performs a two-channel, two-level pulse waveform capture using Segment ARB pulse mode. Both channels use the same pulse timing.

In addition to a SMU bias, the unique feature of this test is that is allows up to 1,000,000 waveform samples (per A/D) to be saved to a comma-separated value (.csv) file on the Model 4200-SCS. There is no spot mean provided by this test. For waveform capture and spot means in a single test, see PMU-ScopeShot. The KITE sheet has a maximum of about 65,000 rows, or samples. This test allows for up to 1,000,000 samples, with down-sampled data for the sheet and graph.

There are two main parameters that control how many waveform samples (rows in the sheet) are returned. MaxFilePoints determines how many samples are captured for each channel (V and I A/D for each channel) and saved to the file. MaxSheetPoints determines how to down-sample the file data so it can be transferred to the sheet and displayed in the graph.

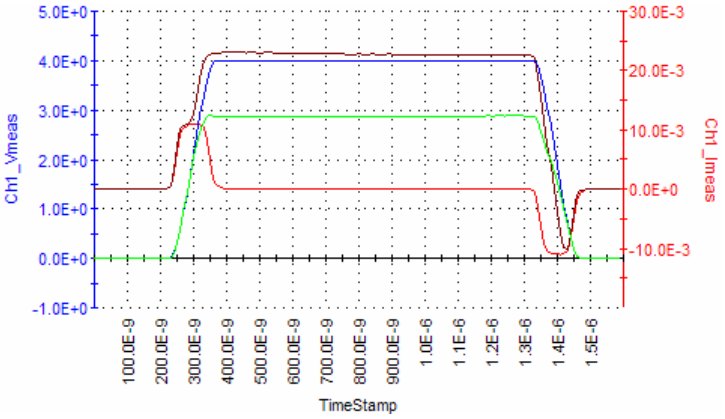
To save the file, set SaveFile = 1. The file location is `C:\s4200\kiuser\export\wavefrm_capture.csv`. To save many tests without having to rename the file for each test, set AppendTimeToFilename = 1, which will append the Windows system date and time.

This test does not provide ranging, LLEC, or set thresholds.

An SMU can be used to provide voltage bias for the substrate. Do not connect the SMU to a DUT drain, source, collector or emitter; unexpected results could occur.

Figure 16-114 shows the setup for this test, and Figure 16-117 shows an example graph.

Figure 16-117
Graph for PMU-Wfm-FileSave



Blue = Ch 1 voltage measure
Red = Ch 1 current measure
Green = Ch 2 voltage measure
Brown = Ch 2 current measure

PMU-ScopeShot

This test performs a two-channel spot mean waveform capture. An SMU can be used to provide voltage bias for the substrate. Figure 16-114 shows the setup for this test. This UTM is similar to the PIV-A and PIV-Q ScopeShot UTMs (see Section 12).

The unique features of this test are independent pulse timing with waveform capture and spot means in a single test. The timing parameters are similar to PMU-IV-Sweep, but with the addition of spot mean measure window with parameters MeasStartGate and MeasStopGate. These parameters are percentages of the pulse top and pulse base, as shown in Figure 16-4.

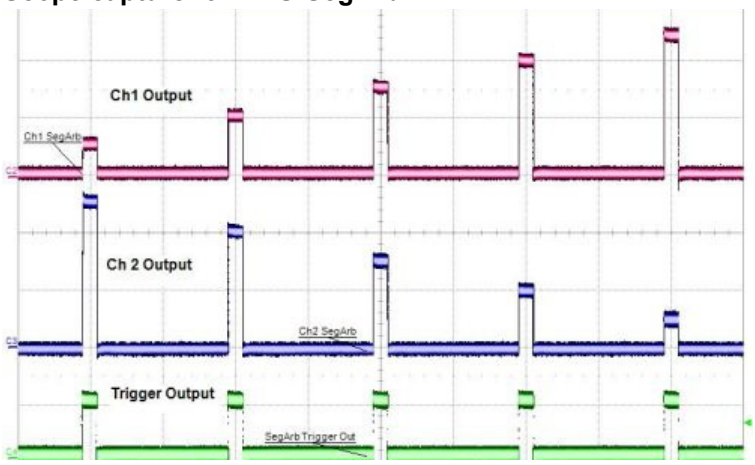
This test supports LLEC, as well as current measure ranging and thresholds. To enable LLEC, set LoadLineGate and/or LoadLine Drain = 1. For limited autoranging, set IRangeGate or IRangeDrain to the desired current range, then set LtdAuto_I_Gate or LtdAuto_I_Drain = 1. For thresholds, set to the desired value. Then set PMUMode = 1. To disable LLEC, ranging, and thresholds, use PMUMode = 0.

An SMU can be used to provide voltage bias for the substrate. Do not connect the SMU to a DUT drain, source, collector or emitter; unexpected results could occur.

PMU-SegArb

This test performs a two-channel amplitude sweep using the Segment ARB pulse mode. Figure 16-114 shows the setup for this test, and Figure 16-118 shows a scope capture of the two channels.

Figure 16-118
Scope capture for PMU-SegArb



The unique feature of this test is configuration and control of Segment ARB pulse mode, which allows for multi-level, multi-pulse waveforms. Figure 16-118 shows the default waveform for this test, which has a five-pulse up-sweep on Ch1 and a down-sweep on Ch2.

To define a waveform, values in several arrays must be entered. For Segment ARB concepts, see seg_arb_define and seg_arb_sequence (in Section 8) and Segment ARB waveforms (in Section 5 of the User's Manual) in KPulse.

This test supports a single sequence, and the number of segments (set by NumSegments) can be from three to 1024. Both channels share timing in the SegTime array. The start and stop voltages are in StartVChn and StopVChn, where "n" refers to 1 or 2 for the PMU channel. The test only provides a waveform capture on all segments in the test; spot means are not available.

The SSRCtrlChn arrays control the solid state relay (SSR) on each channel (see Figure 16-2). These arrays permit the segment-by-segment control of the SSR state (1 = closed, or connected to the output; 0 = open, or disconnected from the output).The SSR is typically used in flash memory testing, but other applications may need to temporarily isolate a channel within a multi-level pulse waveform. Note that the minimum segment time when switching the SSR state is 25 µs. The first value in these arrays must be 1.

Note that there is trigger output control on a segment-by-segment basis by setting values in SegTrigOut. This trigger signal is output on the Trig Out connector (see Figure 16-1). This test does not provide LLEC or set thresholds.

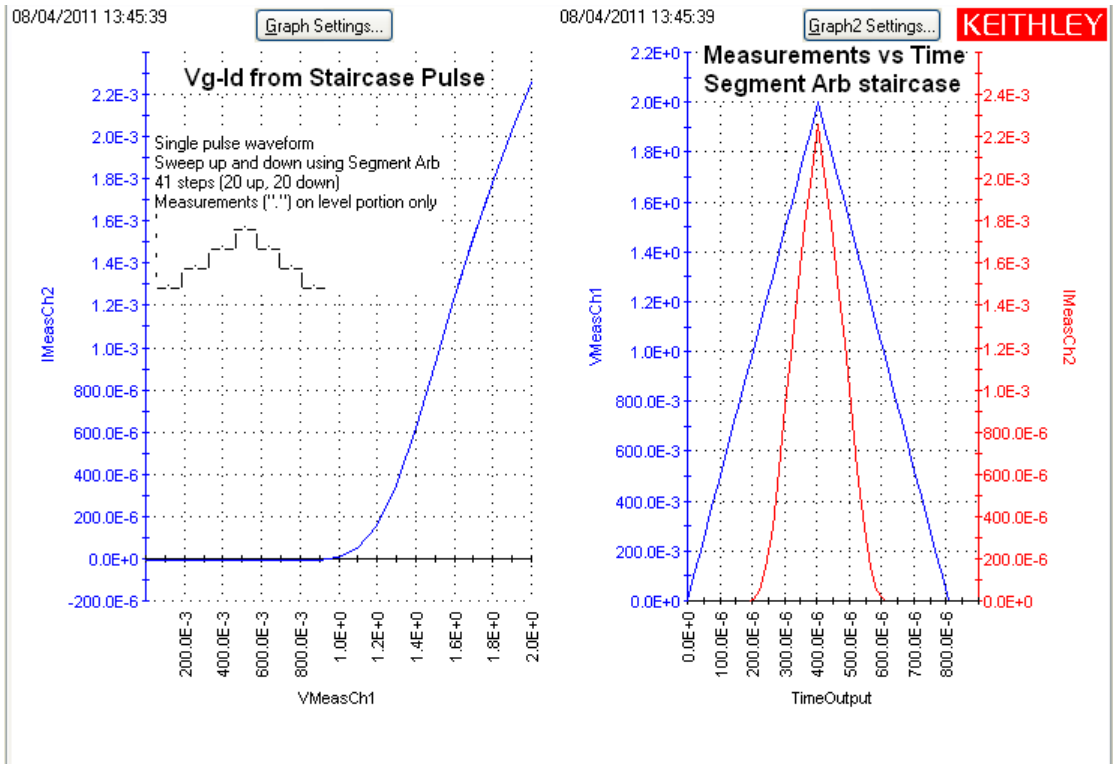
An SMU can be used to provide voltage bias for the substrate. Do not connect the SMU to a DUT drain, source, collector, or emitter; unexpected results could occur.

PMU-SegArb-B

This test is similar to the PMU-SegArb test, but with two differences. The measurements for both channels of the test can be either spot means or sample (waveform). The second difference is that the measurements can be enabled or disabled on a segment basis.

This allows for tests that do a fast staircase sweep. The example shown in Figure 16-120 is an up and down staircase sweep. The left graph shows the I-V results, which is an Vg-Id curve on a transistor. The right graph shows the voltage (blue) and current (red) waveforms versus time. Note that the measurements are only taken on the tops of the pulses; the staircase pulse transitions are not measured.

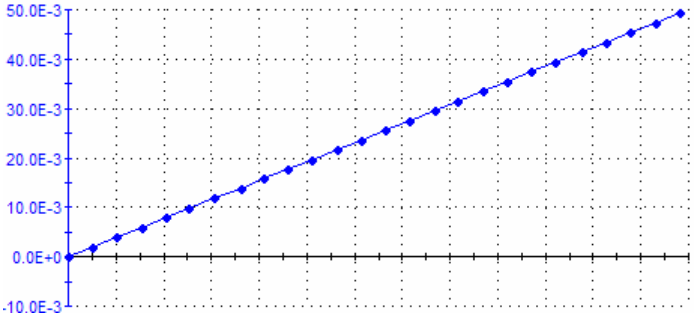
Figure 16-119
Graphs for Segment ARB test



PMU-1Ch-Sweep

This test performs a one-channel spot mean amplitude sweep. Figure 16-114 shows the setup for this test, and Figure 16-120 shows an example graph of the IV measurements on a resistor.

Figure 16-120
Graph for PMU-1Ch-Sweep



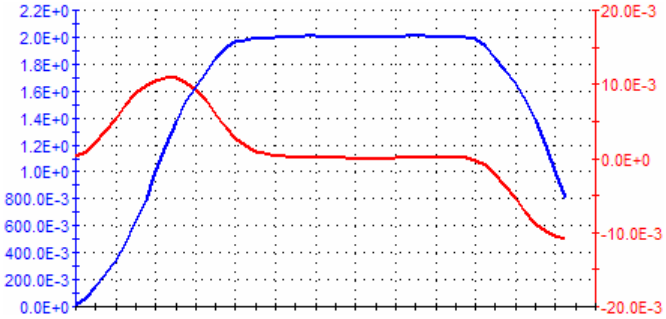
The purpose of this test is a code example for an amplitude sweep of a single PMU channel and return spot means for both I and V. This module allows adjustment of the spot mean start and stop percentages, but does not support ranging or thresholds. LLEC control is available with the LoadLineCh1 parameter.

See Section 6 in the User's Manual for an introduction to programming the Model 4200-SCS using KULT and Section 8 of this Reference Manual for advanced information and card programming command reference.

PMU-1Ch-Wfm

This test performs a one-channel waveform capture. Figure 16-114 shows the setup for this test, and Figure 16-121 shows an example of voltage and current measurements versus time.

Figure 16-121
Graph for PMU-1Ch-Wfm



Blue = Voltage measurements
Red = Current measurements

The purpose of this test is a code example for an amplitude sweep of a single PMU channel and returning waveform capture (signal versus time) for I and V. This module allows adjustment of pre-pulse and post-pulse waveform capture percentages, but does not support ranging or thresholds. LLEC control is available with the LoadLineCh1 parameter.

PMU-SMU-IV-Sweep

This test to perform an IV sweep uses the output relays of the instrument cards to switch the SMU and PMU outputs to the device terminals. Figure 16-115 shows the test setup.

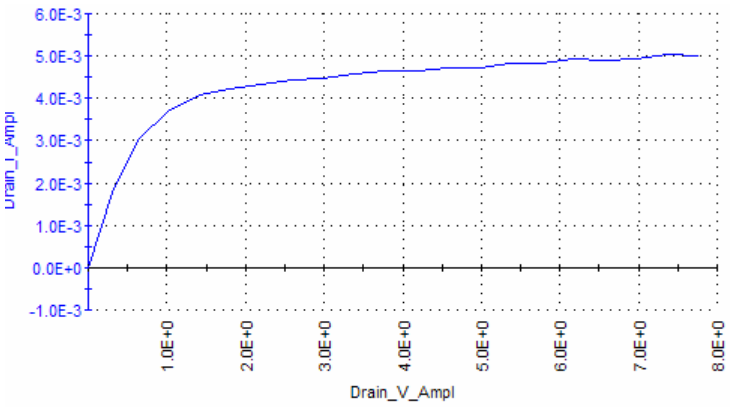
DC bias (SMU1) or a pulse train (PMU Ch1) is typically applied to the gate of the MOSFET. A pulse amplitude sweep (PMU Ch2) or voltage sweep (SMU2) is typically performed on the drain.

The unique feature of this test is that it provides an IV sweep using the PMU and then two SMUs sequentially, without recabling. Figure 16-115 shows the interconnect, which requires adapters included with the 4200-PMU-PROBER-KIT. Note that this test does not require or support the use of Model 4225-RPMs. ExecMode = 0 will perform the PMU Pulse IV sweep, then the SMU DC IV sweep. ExecMode = 1 will perform the PMU PulseIV test. ExecMode = 2 will perform the SMU DC IV test. This test also provides unique pulse timing for Ch1 and Ch2.

Because both SMU and PMU tests are supported, both pulse timing and measure parameters, as well as SMU measure parameters, must be provided. The train (bias) uses BaseVCh1 and AmplVCh1. The amplitude sweep values (StartVCh2, StopVCh2, StepVCh2) are used for both the PMU and SMU sweep. The baseV applies only to the PMU output. There is no SMU bias available during the PMU test.

Figure 16-122 shows an example graph.

Figure 16-122
Graph for PMU-SMU-IV-Sweep



NVM_Examples project

This project has tests for NAND Flash, phase change memory (PCRAM) and ferroelectric memory (FeRAM). It uses two SMUs, one 4225-PMU and two 4225-RPMs. For more information, see the NVM Application Note link on the Applications page of the 4200 Complete Reference.

PMU-Flash-NAND project

This project is used to test flash memory. The tests for this project are summarized in Table 16-17. Figure 16-123 shows the project navigator for the PMU-Flash-NAND project.

See How to perform a flash memory test on my device in Section 3 of the User's Manual for details on flash memory testing.

Figure 16-123
Project navigator for the PMU-Flash-NAND project

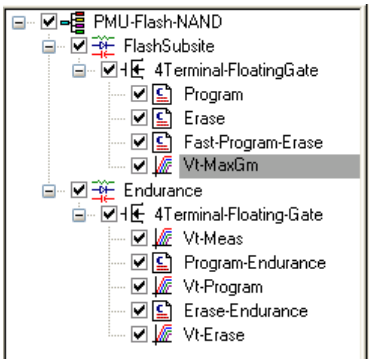
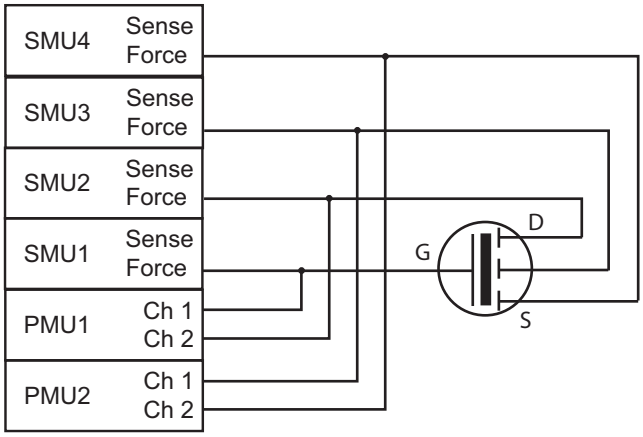


Figure 16-124 shows the test setup for the PMU-Flash-NAND project. A matrix to control switching is not used. Instead, the test uses the output relays of the instrument cards to switch the SMU and PMU outputs to the device terminals.

Figure 16-124
Test setup for the PMU-Flash-NAND project



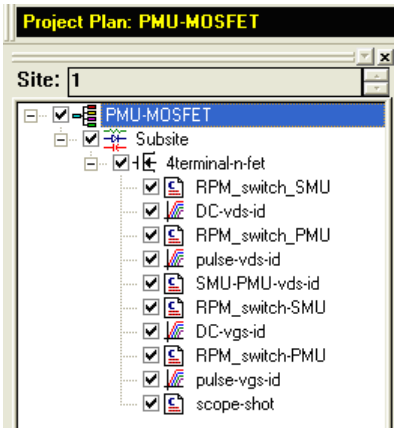
PMU-MOSFET project

The PMU-MOSFET project contains test modules for making both DC and pulsed IV measurements on a MOSFET. In particular, these tests are used to generate DC and pulsed IV drain family of curves and gate voltage versus drain current measurements on a MOSFET. This project relies on the automatic mode switching of the RPM built into the ITMs. This automatic switching uses the type of test (SMU, PMU) to change the mode of the RPM to match (Blue LED for SMU, Green LED for Pulse). Before using the automatic switching of RPMs for the first time, follow the KCON procedure to "Update RPM Configuration" (see Tools > Update DC Preamp and RPM Configuration in Section 7).

Another one of the user modules, SMU-PMU-vds-id, combines the SMU (DC) and PMU (pulsed IV) measurements in one test. Finally, the UTM, scope-shot, enables the user to perform a transient I-V measurement with spot means on a MOSFET.

Figure 16-125 shows the project navigator for the project. All the tests for this project are summarized in Table 16-17.

Figure 16-125
Project navigator for the PMU-Flash-NAND project



The hardware configuration for generating all of the tests (except SMU-PMU-vds-id) is shown in Figure 16-126. SMU1 and RPM 1 are connected to the gate of the transistor, and SMU2 and RPM 2 are connected to the drain of the transistor. The source terminal of the transistor is connected to the LO of the PMU, which is the outside shield of the coax cable on the output of the RPM.

Figure 16-126
Using RPMs to test a MOSFET

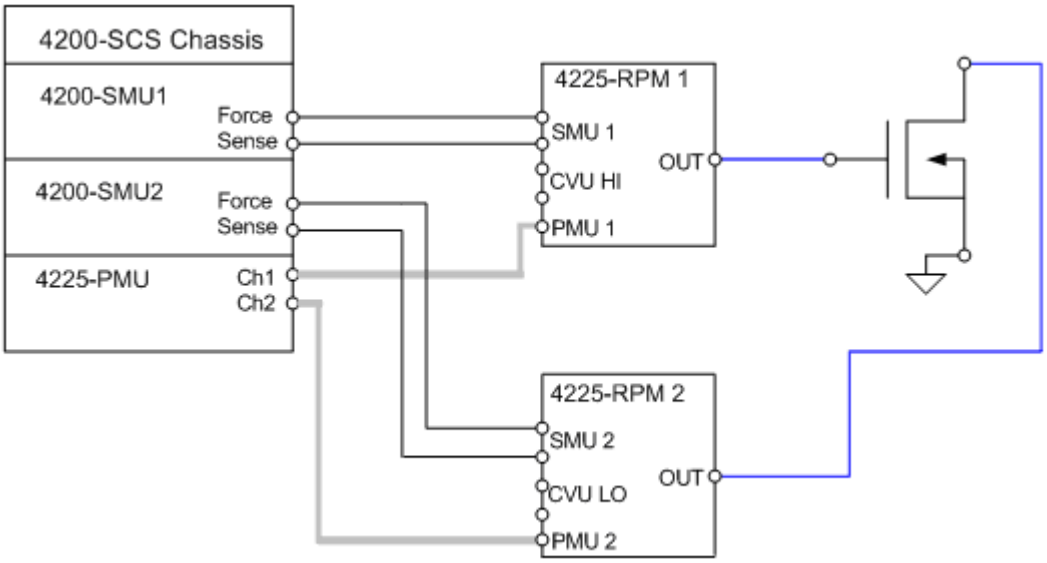
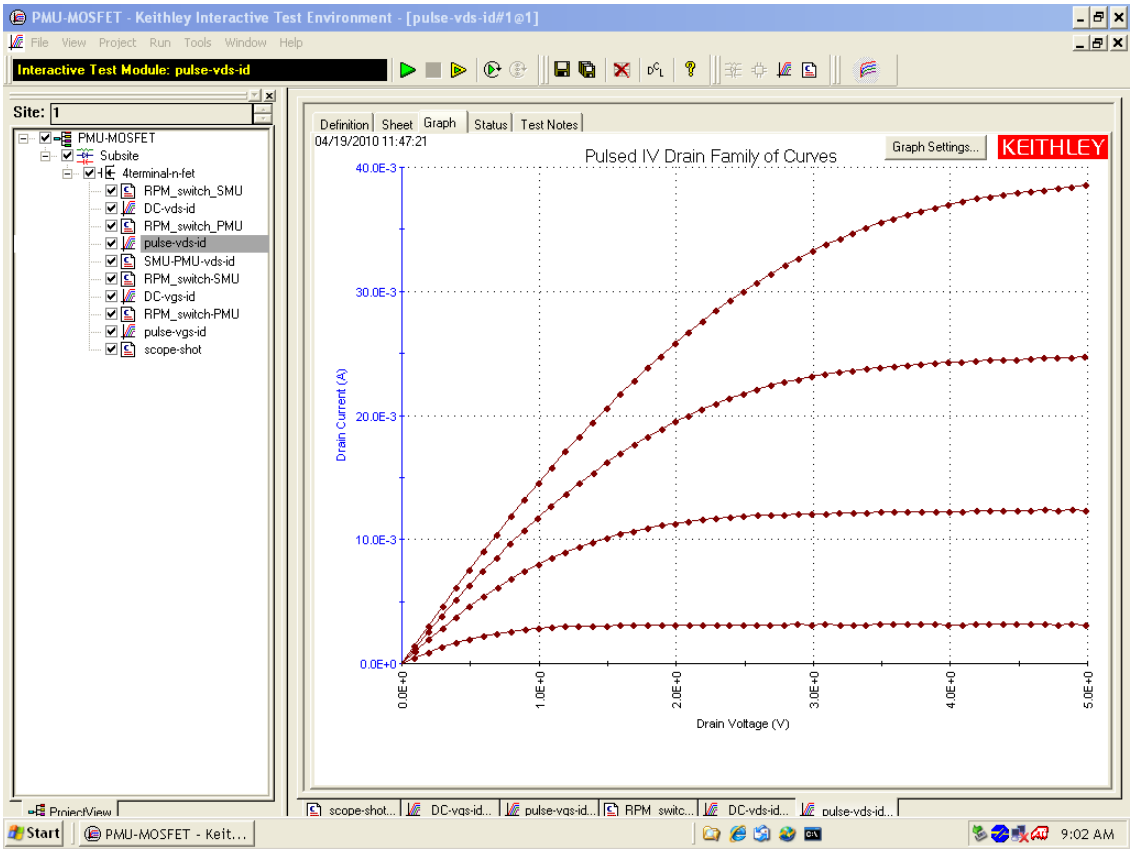


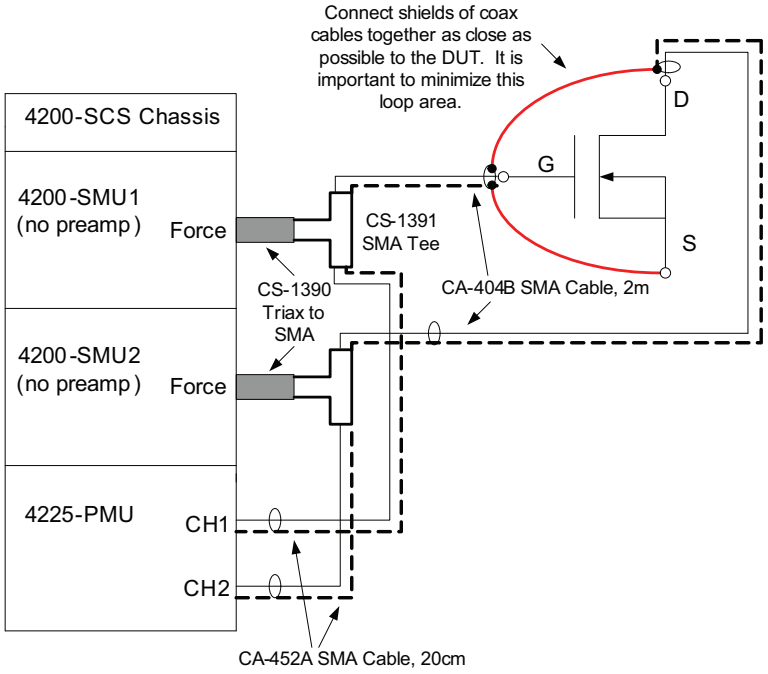
Figure 16-127 shows a pulse IV drain family of curves generated by the pulse-vds-id ITM, using the PMU to perform measurements.

Figure 16-127
Family of curves for pulse_vds-id



The SMU-PMU-vds-id UTM uses the output relays and not the RPMs to switch the SMU and PMU outputs to the MOSFET terminals. Figure 16-128 shows the test setup to perform switching between the SMUs and PMU channels using the output relays on the instrument cards. In this configuration, the SMU preamplifiers and the RPMs cannot be connected in the test circuit.

Figure 16-128
MOSFET test setup (uses output relays for switching)



Cabling from Two 4200-SMUs and 4225-PMU to Three-Terminal MOSFET:

Keithley P/N	QTY	Description
CA-404B	2	SMA to SMA Cable, 2m (supplied with PMU)
CS-1390	2	Male Triax to Female SMA Adaptor (guard removed)*
CS-1391	2	SMA Tee F-M-F*
CA-452A	2	SMA to SMA cable, 20cm*
N/A	2	Jumper Wires to Connect Shields (user supplied)

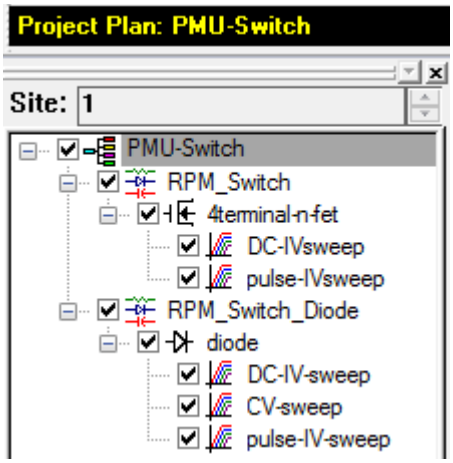
*included in Model 4200-PMU-PROBER-KIT

PMU-Switch project

This project gives two separate examples of switching the PMU to the device under test. The first example switches both the SMU and PMU outputs to the terminals of a MOSFET using the RPM. The second example shows how to switch the SMUs, CVUs, and PMU output terminals through the RPM to a diode. The tests for this project are summarized in Table 16-17.

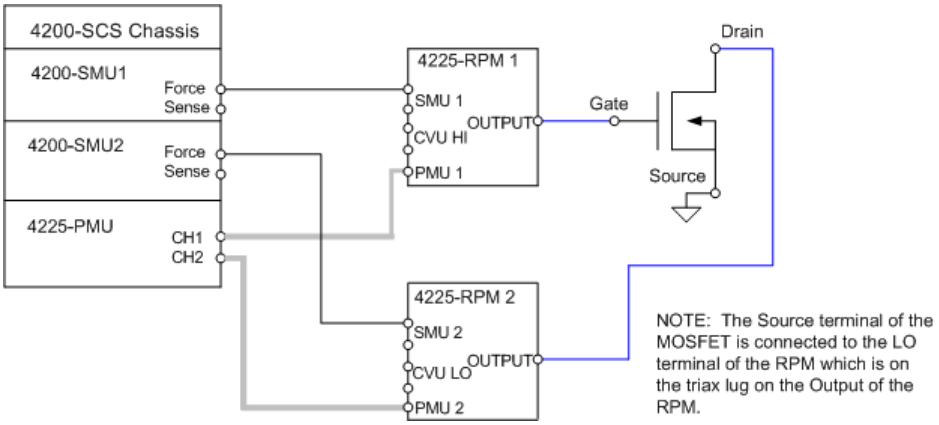
Figure 16-129 shows the project navigator for this project.

Figure 16-129
Project navigator for the PMU-Switch project



The hardware configuration for the MOSFET test is shown in Figure 16-130.

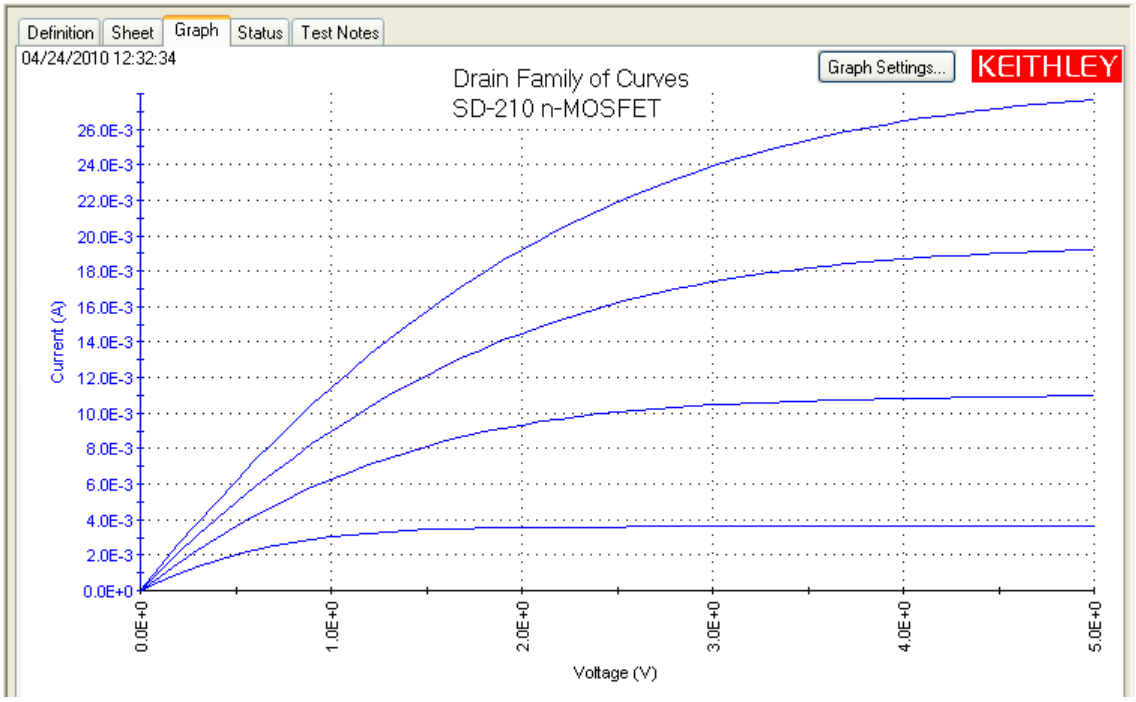
Figure 16-130
Testing a MOSFET using RPMs



The first two test modules are used for switching and measuring the DC and pulsed I-V drain family of curves on a MOSFET. The ITM automatically controls the RPM, putting the RPM into the correct mode: SMU mode (blue LED) for the SMU test and PMU mode (green LED) for the pulse test. Before using the automatic switching of RPMs for the first time, follow the KCON procedure to "Update RPM Configuration" (see Tools > Update DC Preamp and RPM Configuration in Section 7).

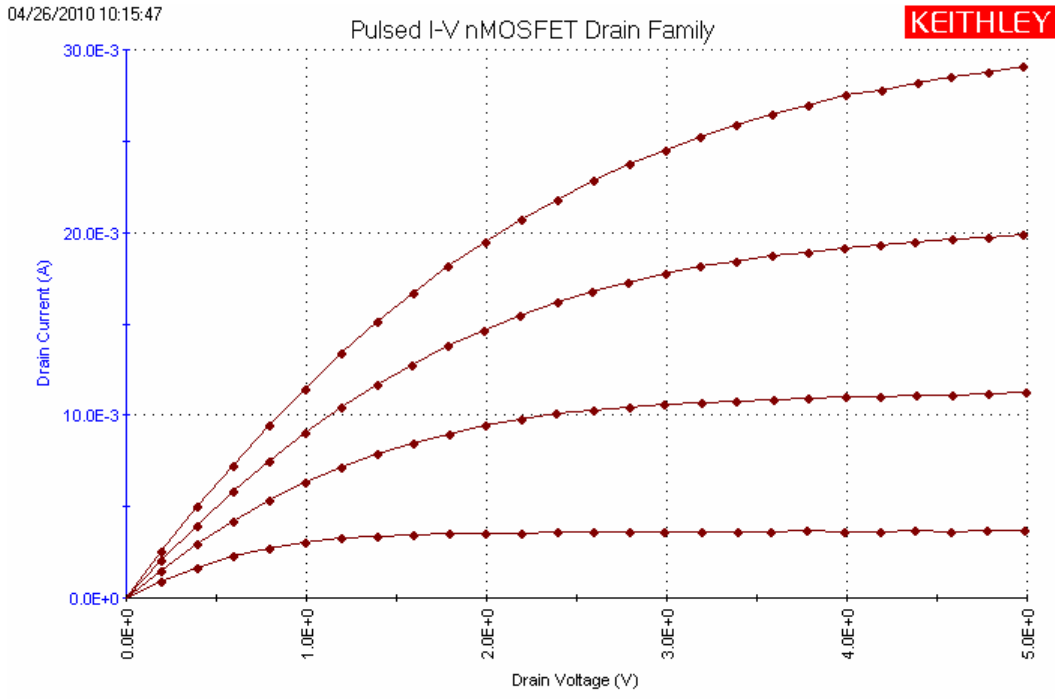
Figure 16-131 shows a family of curves generated by the DC-IVsweep test.

Figure 16-131
Family of curves for DC-IVsweep



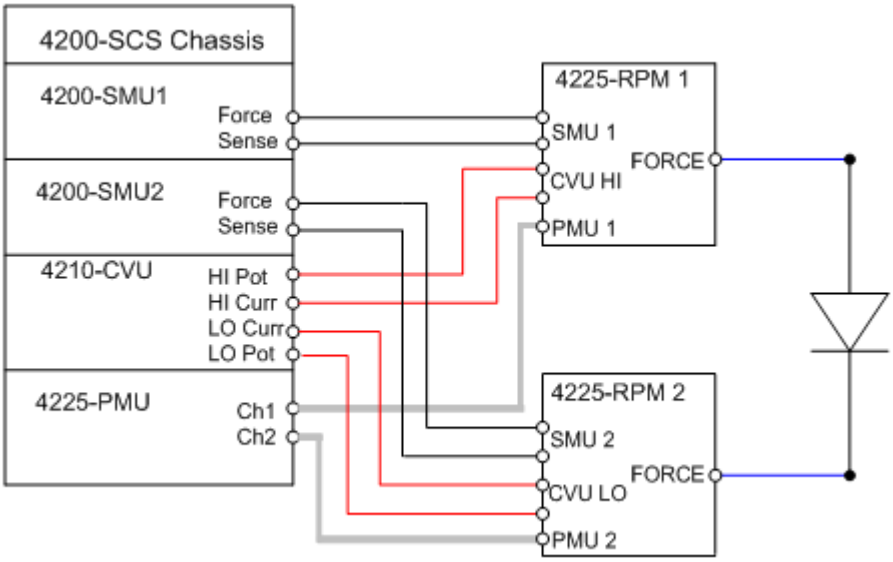
The results of generating a pulsed IV drain family of curves using the pulse-IVsweep ITM are shown in Figure 16-132.

Figure 16-132
Family of curves for pulse-IVsweep



The next example in the PMU-Switch project uses the ITMs to automatically switch the RPM mode to route either the SMUs, the CVU, or the PMU channels to a diode using RPMs. Shown in Figure 16-133 are the hardware connections from the Force output terminals of the two RPMs to the diode. The output of the RPM has both Sense and Force terminals that can be used if a four-wire measurement is necessary to eliminate lead resistance from affecting measurement accuracy. It is important to keep the connections from the output of the RPM to the device as short as possible. Automatic switching of the RPM in CVU mode measures using two-wire measurements.

Figure 16-133
Test setup using the RPM as a switch to test a diode



chargepumping user library

Charge pumping (CP) is a useful technique for understanding gate stack behavior, which is increasingly important as high κ films become more commonly used for transistor gates. CP characterizes interface and charge-trapping phenomena. The change in the CP results can be used to determine the amount of degradation caused by typical reliability test methods, employing either DC or pulsed stress:

- Hot carrier injection (HCI)
- Negative bias temperature instability (NBTI)
- Time-dependent dielectric breakdown (TDDB)

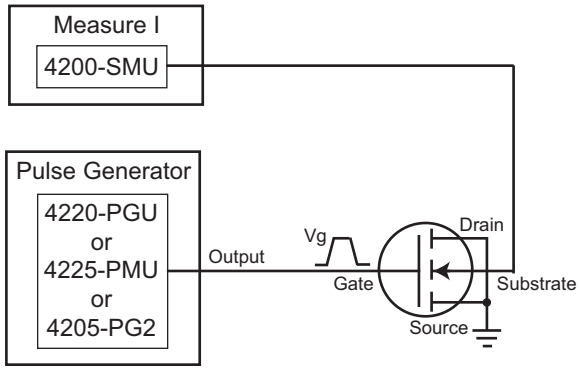
Pulsed voltage provides a key capability for investigating inherent material, interface, and reliability properties of high κ films and devices based on these new films. Pulsing a voltage while simultaneously measuring the DC current is the basis for charge pumping, which is valuable for measuring inherent charge trapping.

Used in conjunction with DC or pulsed stress, CP can also study charge trapping, as well as new charge creation on the high κ-Si interface and within the high κ film. Pulsed stress also provides a stress method that better mimics actual stresses seen by the in-circuit devices, which is useful for various device reliability tests, including NBTI, TDDB, and HCI.

The Model 4200-SCS includes a user library that enables the user to make charge pumping measurements. The user modules in the library use the Model 4200 SMU to measure the charge pumping current (I_{CP}), and the gate is pulsed using either the Models 4225-PMU, 4220-PGU, or 4205-PG2.

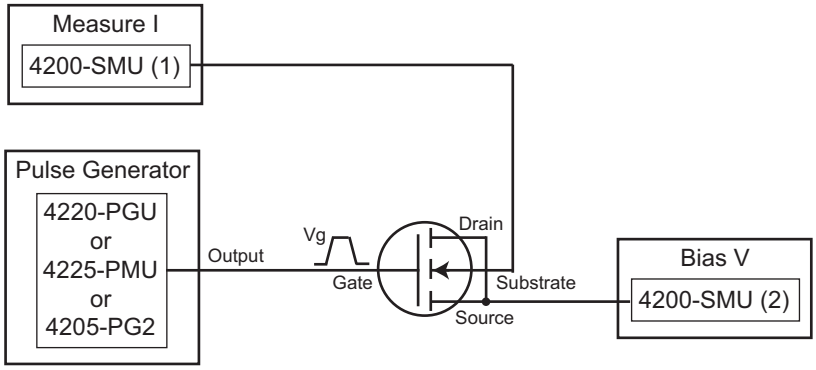
The charge pumping configuration shown in Figure 16-134 uses a single SMU and a pulse generator. The source and drain of the transistor are connected to ground, while the gate is pulsed with a fixed frequency and amplitude. The substrate is connected to an SMU that is used to measure the current through the gate (I_{CP}).

Figure 16-134
Charge pumping—hardware setup block diagram using one SMU



The charge pumping configuration shown in Figure 16-135 uses two SMUs and a pulse generator. The gate is pulsed with a fixed frequency and amplitude. One SMU measures I_{CP} and the other SMU applies a DC bias voltage to the source and drain terminals.

Figure 16-135
Charge pumping—hardware setup block diagram using two SMUs



The chargepumping user library contains several user modules for performing common charge pumping tests. Table 16-18 lists and briefly describes the user modules. See Charge pumping user module descriptions for details of the user modules.

Table 16-18
Charge pumping user modules

User module	Description
AmplitudeSweep	Pulse amplitude is swept while the SMU base voltage is kept constant. The charge pumping current (I_{CP}) is measured as a function of the pulse amplitude voltage. The single SMU setup shown in Figure 16-134 is used with this user module.
AmplitudeSweep_2SMU	Same as the AmplitudeSweep user module, except that it uses a second SMU to apply a DC bias voltage to the source/drain terminals. The dual SMU setup shown in Figure 16-135 is used with this user module.

Table 16-18 (continued)
Charge pumping user modules

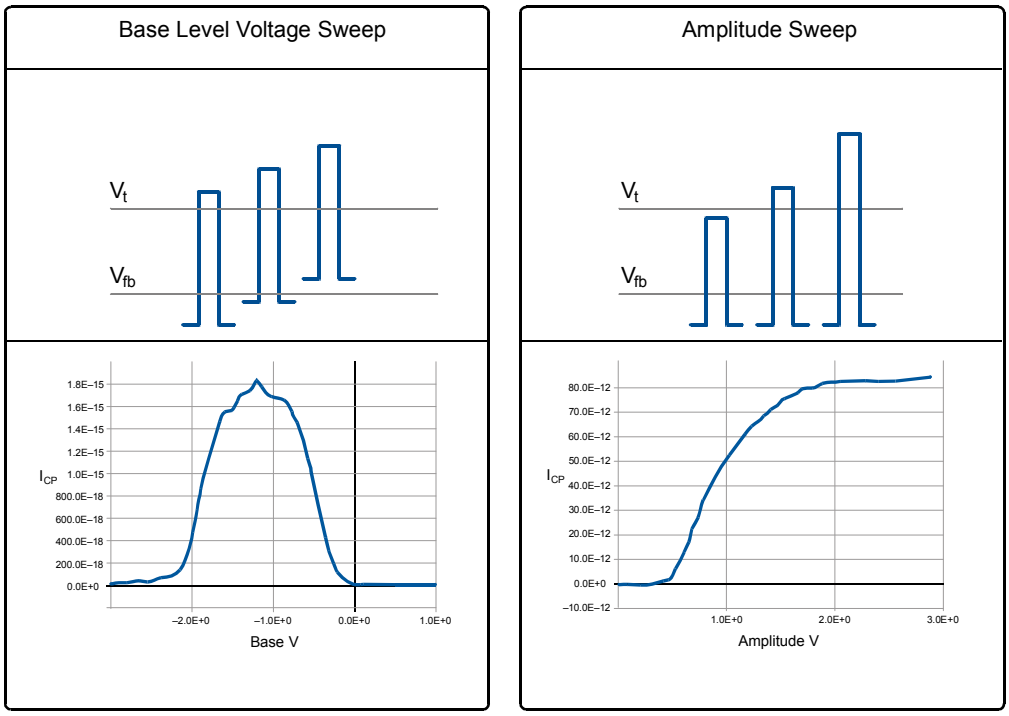
User module	Description
BaseSweep	The base voltage of the waveform is swept by an SMU while the amplitude of the pulse is kept constant. The resulting charge pumping (I_{CP}) is measured as a function of the base voltage. The single SMU setup shown in Figure 16-134 is used with this user module.
BaseSweep_2SMU	Same as BaseSweep user module, except it uses a second SMU to apply a DC bias voltage to the source/drain terminals. The dual SMU setup shown in Figure 16-135 is used with this user module.
FallTimeLinearSweep	Performs a linear sweep of the falling transition time of the pulse. Charge pumping current (I_{CP}) is measured and graphed as a function of the fall time. The single SMU setup shown in Figure 16-134 is used with this user module.
FreqFactorSweep	With the pulse amplitude, offset voltage, and rise/fall time kept constant, the charge pumping current (I_{CP}) is measured as a function of a multiplier factor controlled frequency sweep of the test frequency. The single SMU setup shown in Figure 16-134 is used with this user module.
FreqLinearSweep	With the pulse amplitude, offset voltage, and rise/fall time kept constant, the (I_{CP}) is measured as a function of a linear sweep of the test frequency. The single SMU setup shown in Figure 16-134 is used with this user module.
RiseTimeLinearSweep	Performs a linear sweep of the rising transition time of the pulse. Charge pumping (I_{CP}) is measured as a function of the rise time. The single SMU setup shown in Figure 16-134 is used with this user module.

From the user modules in the chargepumping user library, a project has been created to perform several charge pumping tests. See chargepumping user library for information on using this project. The following procedure explains how to create a new UTM to perform charge pumping measurement.

Procedure to perform charge pumping measurements

1. Using Table 16-18, choose the desired test method. See Charge pumping user module descriptions for details on the user modules.
2. Connect the transistor as shown in Figure 16-92 (if using one SMU) or Figure 16-93 (if using two SMUs). For the one-SMU configuration, make sure the source and drain are connected to ground.
3. From KITE, create a new UTM in a project plan.
4. In the Definition tab for the UTM, select the chargepumping user library, and then select the desired user module.
5. In the Definition tab, input the appropriate test parameters.
6. Save and then execute the UTM. The UTM pulses the gate with a train of pulses. While pulsing, the UTM will measure the DC substrate current with the SMU. This is the charge pumping current (I_{CP}). Plot the data in the graph.
7. After one measurement is finished, change the pulse characteristics and measure again using either the Append Data run button or by creating a new UTM and then running it. Pulse parameters are changed based on the type of sweep. See Figure 16-136 for examples of base and amplitude sweeps. These two plots, Base Level Voltage Sweep and Amplitude Sweep, show I_{CP} measured as a function of the pulse voltage.

Figure 16-136
Two types of sweeps for charge pumping



8. From the curve, the interface trap density (N_{it}) can be extracted, based on the following equation:

$$N_{it} = \frac{I_{CP}}{qfA}$$

Where: N_{it} = Interface trap charge density (cm^{-2})
I_{CP} = Charge pumping current (A)
f = Test frequency (Hz)
q = Electron charge (1.6022×10^{-19} C)
A = Channel area (cm^2)

The trap density as a function of band bending can also be extracted from the following equation:

$$D_{it} = \frac{I_{CP}}{qfA\Delta E}$$

Where: D_{it} = Interface trap charge density ($cm^{-2}eV^{-1}$)
I_{CP} = Charge pumping current (A)
f = Test frequency (Hz)
q = Electron charge (1.6022×10^{-19} C)
A = Channel area (cm^2)
ΔE = Difference between the inversion Fermi level and the accumulation Fermi level

To simplify this procedure, a project has already been created to perform these measurements. The project is called “chargepumping,” and is located on the Model 4200-SCS in the following folder:

```
C:\S4200\kiuser\Projects\_Pulse
```

The chargepumping project contains UTMs for the user modules in the chargepumping user library. See the for detailed information about the chargepumping project.

Charge pumping user module descriptions

The chargepumping user library contains modules required to characterize interface and charge-trapping phenomena. The modules contained in the charge pumping user library are listed in Table 16-18, with detailed information following the table.

AmplitudeSweep

Description AmplitudeSweep is a charge pumping routine that performs a linear sweep of the pulse amplitude, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-19). Table 16-20 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full 0 to 100 percent transition times that are used to program the pulse.

Connection This procedure requires connection of the appropriate pulse channel to the gate of the DUT and the substrate/well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to Figure 16-92.

Table 16-19
Inputs for AmplitudeSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator output channel, 1 or 2.
SubSMU	The SMU for the substrate/well. This can be SMU1 up to the maximum number of SMUs in the system.
StartVampl	Starting pulse amplitude (V). This can be set from -40 V to +40 V.
StopVampl	Stopping pulse amplitude voltage for the sweep (V). This can be set from -40 V to +40 V.
StepVampl	Pulse step size for the amplitude sweep (V). This can be set from -40 V to +40 V.
PulseOffset	Offset, or base, of the pulse (V). The SMU can be set from -40 V to +40 V.
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.

Table 16-19 (continued)
Inputs for AmplitudeSweep

Input	Description
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.
DutyCyclePercent	Pulse duty cycle (in percent). This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
PulseAmpl_size Icp_size Qcp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-20
Outputs for AmplitudeSweep

Output	Description
PulseAmpl	The array of pulse amplitudes used.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

AmplitudeSweep_2SMU

Description AmplitudeSweep_2SMU is a charge pumping routine that performs a linear sweep of the pulse amplitude, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse measure unit or a pulse-only card as well as two Model 4200-SCS SMUs.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-21). Table 16-22 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full 0 to 100 percent transition times that are used to program the pulse.

Connection This procedure requires connection of the appropriate pulse channel to the gate of the DUT and the substrate/well to the Model 4200-SCS SMU Force. The other DUT

pins should be connected to a second SMU. For detailed connection information, refer to Figure 16-93.

Table 16-21
Inputs for AmplitudeSweep_2SMU

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator output channel, 1 or 2.
SubSMU	The SMU for the substrate/well. This can be SMU1 up to the maximum number of SMUs in the system.
sdSMU	The SMU for the subsite and drain terminals. Specify as SMU1, SMU2, and so on.
StartVampl	Starting pulse amplitude (V). This can be set from -40 V to +40 V.
StopVampl	Stopping amplitude voltage for the sweep (V). This can be set from -40 V to +40 V.
StepVampl	Step size for the amplitude sweep (V). This can be set from -40 V to +40 V.
sdBias	Amplitude of the voltage bias applied by the sdSMU to the source and drain. This can be set from -80 V to + 80 V.
PulseOffset	Offset, or base, of the pulse (V). This can be set from -40 V to +40 V.
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU1 integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed, or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
PulseAmpl_size Icp_size Qcp_size sdSMUCompliance_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-22
Outputs for AmplitudeSweep_2SMU

Output	Description
PulseAmpl	The array of pulse amplitudes used.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

BaseSweep

Description BaseSweep is a charge pumping routine to perform a linear sweep of the pulse base or offset, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-23). Table 16-24 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to Figure 16-92

Table 16-23
Inputs for BaseSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator output channel, 1 or 2.
SubSMU	The SMU for the substrate/well. This can be SMU1 up to the maximum number of SMUs in the system.
StartVBase	Starting pulse base (V). This can be set from -40 V to +40 V.
StepVBase	Step size for the base sweep (V). This can be set from -40 V to +40 V.
StopVBase	Stopping amplitude voltage for the sweep (V). This can be set from -40 V to +40 V.
PulseAmplitude	This sets pulse amplitude (V) from -40 V to +40 V.
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.

Table 16-23 (continued)
Inputs for BaseSweep

Input	Description
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E-6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
BaseV_size Icp_size Qcp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-24
Outputs for BaseSweep

Output	Description
BaseV	The array of pulse base voltages.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

BaseSweep_2SMU

Description BaseSweep_2SMU is a charge pumping routine to perform a linear sweep of the pulse base or offset, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU. A second SMU is used to apply a DC voltage bias to the source and drain.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-23). Table 16-24 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other

DUT pins should be connected to other Model 4200-SMU. For detailed connection information, refer to Figure 16-93.

Table 16-25
Inputs for BaseSweep_2SMU

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator output channel, 1 or 2.
SubSMU	The SMU for the substrate/well. This can be SMU1 up to the maximum number of SMUs in the system.
sdSMU	The SMU for the subsite and drain terminals. Specify as SMU1, SMU2, and so on.
StartVBase	Starting pulse base (V). This can be set from -40 V to +40 V.
StepVBase	Step size for the base sweep (V). This can be set from -40 V to +40 V.
StopVBase	Stopping amplitude voltage for the sweep (V). This can be set from -40 V to +40 V.
sdBias	Amplitude of the voltage bias applied by the sdSMU to the source and drain. This can be set from -80 V to + 80 V.
PulseAmplitude	This sets pulse amplitude (V) from -40 V to +40 V.
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E-6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU1 integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
BaseV_size Icp_size Qcp_size sdSMUCompliance_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-26
Outputs for BaseSweep_2SMU

Output	Description
BaseV	The array of pulse base voltages
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

FallTimeLinearSweep

Description The FallTimeLinearSweep is a charge pumping routine that performs a linear sweep of the falling transition time of the pulse, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-27). Table 16-28 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to the Figure 16-92.

Table 16-27
Inputs for FallTimeLinearSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator channel, 1 or 2.
SubSMU	The SMU number. This can be SMU1 to the maximum number of SMUs in the system.
StartFallTime	Starting transition fall time for sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
StopFallTime	Stopping transition fall time for sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
StepFallTime	Stepsize for transition fall time sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.

Table 16-27 (continued)
Inputs for FallTimeLinearSweep

Input	Description
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseAmplitude	Amplitude of pulse (V). This can be set from -40 V to +40 V.
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.
PulseOffset	Offset, or base, of the pulse (V). This can be set from -40 V to +40 V.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
FallTimeSize Icp_size Qcp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-28
Outputs for FallTimeLinearSweep

Output	Description
FallTransTime	The array of fall transition times used.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

FreqFactorSweep

Description

The FreqFactorSweep is a charge pumping routine that performs a multiplier-factored frequency sweep of the pulse, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-29). Table 16-30 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to Figure 16-92.

Table 16-29
Inputs for FreqFactorSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator channel, 1 or 2.
SubSMU	The SMU number. This can be SMU1 to the maximum number of SMUs in the system.
StartFreq	Starting pulse frequency for the sweep. This can be set from 1 to 20 E6 (20 MHz).
FreqMultFactor	Multiplier factor to control step size. Next Freq = Previous frequency * FreqMultFactor. Use factors > 1 for sweeping to higher frequencies. Use factors < 1 for sweeping to lower frequencies.
NumPoints	Number of points in the frequency sweep.
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to 100%), not 10% to 90%.
PulseAmplitude	Amplitude of pulse (V). This can be set from -40 V to +40 V.
PulseOffset	Offset, or base, of the pulse (V). This can be set from -40 V to +40 V.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
Frequency_size Qcp_size Icp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-30
Outputs for FreqFactorSweep

Output	Description
Frequency	The array of frequencies used.
Icp	The array of current values measured by the SMU.

Table 16-30 (continued)
Outputs for FreqFactorSweep

Output	Description
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

FreqLinearSweep

Description The FreqLinearSweep is a charge pumping routine to perform a linear sweep of the pulse frequency, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate values for the charge pumping parameters (see Table 16-31). Table 16-32 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to the Figure 16-92.

Table 16-31
Inputs for FreqLinearSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator channel, 1 or 2.
SubSMU	The SMU number. This can be SMU1 to the maximum number of SMUs in the system.
StartFreq	Starting pulse frequency for sweep (Hz). This can be set from 1 Hz to 20 MHz (20 E6).
StopFreq	Stopping pulse frequency for sweep (Hz). This can be set from 1 Hz to 20 MHz (20 E6).
StepFreq	Pulse frequency step size for sweep (Hz). This can be set from 1 Hz to 20 MHz (20 E6).
PulseRiseTime	Transition rise time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
PulseAmplitude	Amplitude of pulse (V). This can be set from -40 V to +40 V.
PulseOffset	Offset, or base, of the pulse (V). This can be set from -40 V to +40 V.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.

Table 16-31 (continued)
Inputs for FreqLinearSweep (continued)

Input	Description
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
Frequency_size Qcp_size Icp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-32
Outputs for FreqFactorSweep

Output	Description
Frequency	The array of frequencies used.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

RiseTimeLinearSweep

Description

The RiseTimeLinearSweep is a charge pumping routine to perform a linear sweep of the rising transition time of the pulse, graphing the resulting charge pumping current measured by a Model 4200-SCS SMU. This routine controls a single channel of a pulse-measure unit or a pulse-only card as well as a Model 4200-SCS SMU.

Make sure to set the appropriate value for the charge pumping parameters (see Table 16-33). Table 16-34 contains the routines output parameters.

The rise time and fall time parameters are the full transition times (zero to 100 percent), not the 10 percent to 90 percent times.

For the 5 V range of the pulse generator, the 10 percent to 90 percent rise times are about 20 percent less than the full zero to 100 percent transition times that are used to program the pulse.

Connection

Using this routine requires connection of the appropriate pulse channel to the gate of the DUT and the substrate or well to the Model 4200-SCS SMU Force. The other DUT pins should be connected to ground. For detailed connection information, refer to the Figure 16-92.

Table 16-33
Inputs for RiseTimeLinearSweep

Input	Description
VPUId	The instrument ID for the pulse card. For Model 4225-PMU, this should be set to PMU1, PMU2, PMU3, and so on. For the Models 4220-PGU or 4205-PG2, this should be set to VPU1, VPU2, VPU3, and so on.
PulseChan	The pulse generator channel, 1 or 2.
SubSMU	SMU number. This can be SMU1 to the maximum number of SMUs in the system.
StartRiseTime	Starting transition rise time for sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
StopRiseTime	Stopping transition rise time for sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
StepRiseTime	Stepsize for transition rise time sweep (s). This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
PulseFallTime	Transition fall time for the pulse. This can be set from 10 E-9 (10 ns) to 33 E-3 (33 ms) for the high-speed range or 50 E-9 (50 ns) to 33 E-3 (33 ms) for the high-voltage range (with 10 ns resolution). This value programs the full transition time (0 to100%), not 10% to 90%.
PulseAmplitude	Amplitude of pulse (V). This can be set from -40 V to +40 V.
PulseFrequency	Pulse frequency. This can be set from 1 Hz to 20 Mhz.
PulseOffset	Offset, or base, of the pulse (V). This can be set from -40 V to +40 V.
DutyCyclePercent	Duty cycle percent. This can be set from 0.001% to 99.9%.
PulseLoad	DUT load or impedance (ohm [Ω]). This can be set from 50 to 1 E6. This value is used to adjust the pulse amplitude sourced by the pulse generator card to compensate for non-50 Ω termination. For example, setting the load = 1 E6 means the pulse generator card will output half the voltage compared to load = 50.
NPLC	SMU integration time in power line cycles. This can be set from 0.01 to 10.
PulseRange	Pulse range. Set this value to 5 (50 Ω load) or 10 (≥1M Ω load) for high speed or to 20 (50 Ω load) or 40 (≥1M Ω load) for high voltage.
LowestIRange	Lowest SMU current measure range used during limited autorange. This can be set from 10 e-12 to 100 e-3.
SMUCompliance	Current limit for the SMU. Set from 10 e-12 to 100 e-3.
RiseTime_size Qcp_size Icp_size	Set to a value that is at least equal to the number of steps in the sweep. All _size values must be equal to each other.

Table 16-34
Outputs for RiseTimeLinearSweep

Output	Description
Frequency	The array of fall transition times used.
Icp	The array of current values measured by the SMU.
Qcp	The array charge values calculated from the Icp values, where: Qcp = Icp/(Frequency)

Appendix A
Creating Project Prompts

In this section:

Key concepts

Project prompts overview

Dialog boxes are available to pause a test sequence with a prompt. These dialog boxes are available as user modules (see Table A-1). The user defines the text message for the prompt. When one of these user modules is run, the test sequence will pause. The test sequence will continue when a button on the dialog box is clicked.

Table A-1
Winulib user library

User module	Description
AbortRetryIgnoreDialog	Pause test sequence with a prompt to Abort, Retry or Ignore
InputOkCancelDialog	Pause test sequence for an input prompt; enter input data (OK) or Cancel
OkCancelDialog	Pause test sequence with a prompt to continue (OK) or Cancel
OkDialog	Pause test sequence with a prompt to continue (OK)
RetryCancelDialog	Pause test sequence with a prompt to Retry or Cancel
YesNoCancelDialog	Pause test sequence with a Yes, No, or Cancel decision prompt
YesNoDialog	Pause test sequence with a Yes or No decision prompt

Using dialog windows

The Winulib user library has user modules for six action/decision dialog boxes and one input dialog box. The dialog box, along with example prompts, are shown in Figure A-1 and Figure A-2. The text message for a prompt is entered by the user into the user module. See Winulib user-library reference in this appendix for details on the user modules.

The Ok dialog box in Figure A-1A has only one button. This dialog box is used to pause a test sequence to make an announcement (for example, Test Finished), or prompt for an action (for example, Connect 590 to DUT). When the **OK** button is clicked, the test sequence continues.

NOTE: *An example using the Ok dialog box is provided in "Dialog box test examples" in this appendix.*

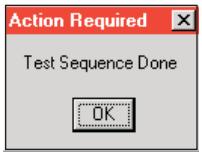
Parameter passing

The rest of the dialog boxes in Figure A-1 and Figure A-2 have two or three buttons. When a button on a dialog box is clicked, a status value (parameter) that corresponds to that action or decision is placed in its **Sheet** tab. When one or more input parameters are entered using the input dialog box (Figure A-2), each parameter is also placed in its **Sheet** tab (data spreadsheet). A parameter value can then be passed into a user-created routine to perform a desired action or operation.

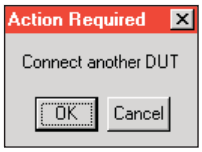
In order to pass parameters, the dialog box user module must be called from another user-created user module that is designed for parameter passing. A parameter that is placed in the **Sheet** tab is passed to a routine in the user-created user module to perform the appropriate operation or action.

NOTE: *An example to demonstrate parameter passing is provided in Dialog box test examples in this appendix.*

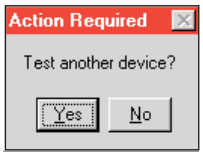
Figure A-1
Decision dialog windows



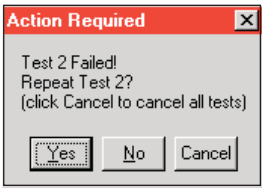
A. OkDialog



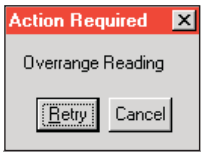
B. OkCancelDialog



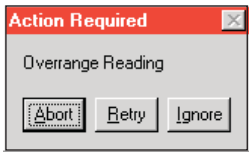
C. YesNoDialog



D. YesNoCancelDialog

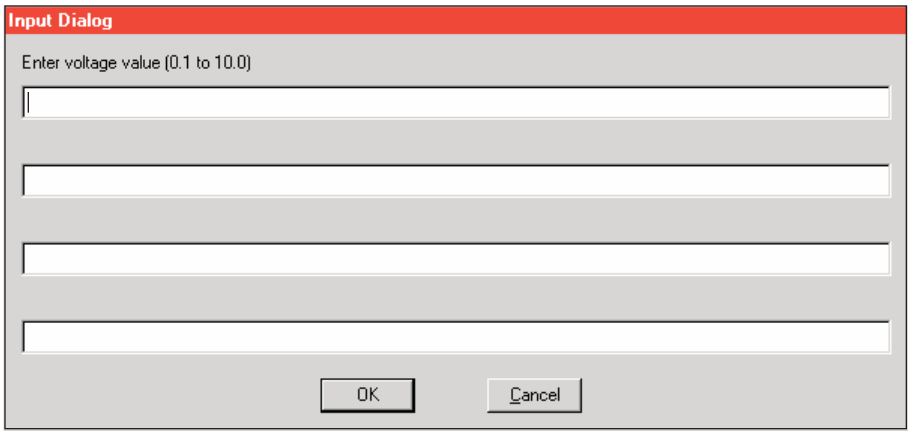


E. RetryCancelDialog



F. AbortRetryIgnoreDialog

Figure A-2
Input dialog window



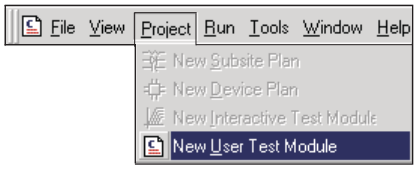
Dialog box test examples

Announce end of test

For this example, a C that uses the Ok dialog user module will be created. This dialog box will announce the end of a test sequence. This UTM can be used in any project, at the end of any test sequence.

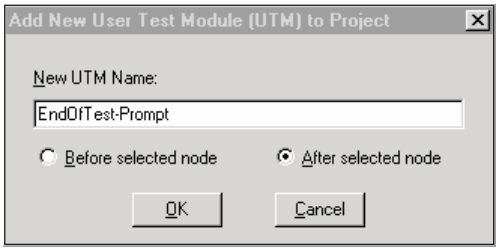
1. In the Project Navigator for any project, click the last interactive test module (ITM) or UTM in a test sequence. This marks the insertion point for the new UTM.
2. From the **Project** menu on the toolbar, select **New User Test Module** (see Figure A-3).

Figure A-3
Project menu



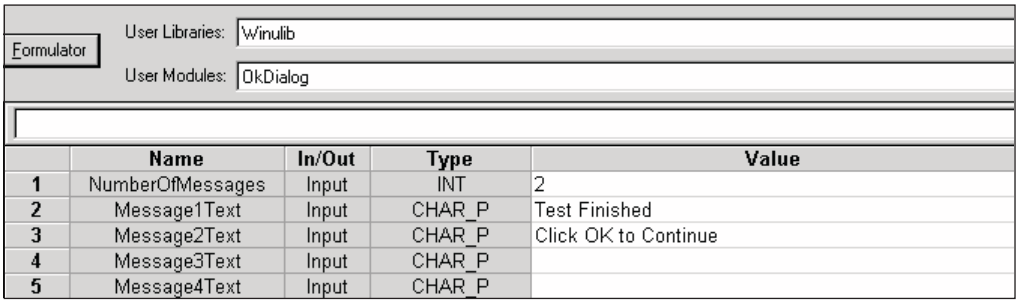
3. In the new UTM window (Figure A-4), type in a name for the UTM. Click **After selected node**, and then click **OK**. This inserts the new UTM at the end of the test sequence.

Figure A-4
New UTM window



4. In the Project Navigator, double-click the new UTM to open it. Select the **Winulib** library, the **OkDialog** module, and configure the user module as shown in Figure A-5.

Figure A-5
New UTM using OkDialog user module



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	2
2	Message1Text	Input	CHAR_P	Test Finished
3	Message2Text	Input	CHAR_P	Click OK to Continue
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

5. From the **File** menu, click **Save** to initialize the new UTM.

When the test sequence is run, the dialog box indicating that the test is finished will appear when it is executed. Press **OK** to continue.

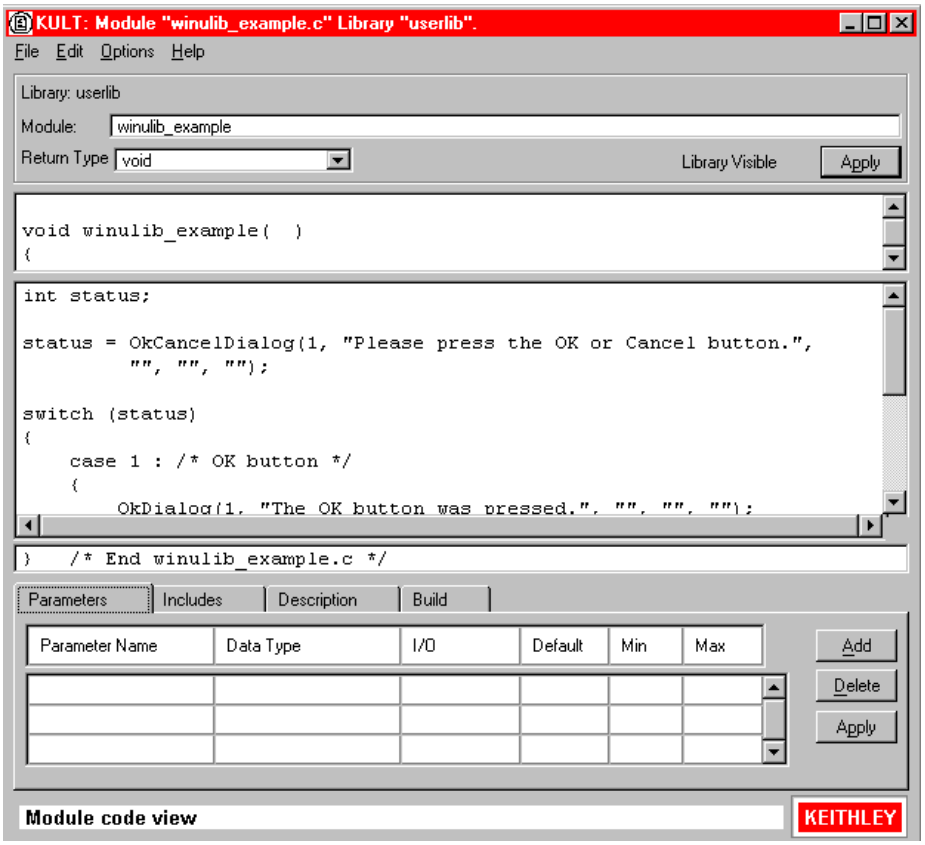
Parameter passing

This example demonstrates how a user-created user module calls **Winulib** user modules (dialog boxes) and how parameters are passed when a dialog box button is pressed.

This example assumes that a user library named **userlib** and user module named **Winulib_example** already exists. Figure A-6 shows the KULT window for the **Winulib_example** user module.

NOTE: *Detailed information on creating a user library and user module are provided in "Keithley CONfiguration Utility (KCON)" in Section 7.*

Figure A-6
KULT window for Winulib_example



Program listing

The complete program listing (C language) for the **Winulib_example** user module is detailed below. This is the program that configures and calls **Winulib** user modules and acts on parameters passed to it when a dialog box button is clicked.

```
int status;
status = OkCancelDialog(1, "Please press the OK or Cancel
          button.", "", "", "");
switch (status)
{
    case 1 : /* OK button */
    {
      OkDialog(1, "The OK button was pressed.", "", "", "");
      break;
    }
    case 2 : /* Cancel button */
    {
      OkDialog(1, "The Cancel button was pressed.", "", "", "");
     break;
    }
    default : /* An error occurred */
    {
      OkDialog(1, "OkCancelDialog() had an error.", "", "", "");
     break;
    }
}
```

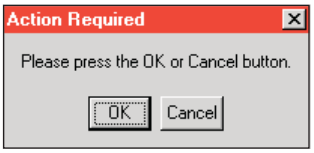
Create a UTM for the Winulib_example user module

Create and initialize a new UTM that uses the **Winulib_example** user module by following steps 2 through 5 of the example, Announce end of test, that appears earlier in this appendix. Ensure that in the UTM you select the **userlib** user library and **Winulib_example** user module.

Executing the test

With the new UTM selected, execute the test by clicking the green **Run** button. The program for **Winulib_example** configures and calls (displays) the OkCancelDialog window as shown in Figure A-7.

Figure A-7
OkCancelDialog window



If you click the **OK** button, the parameter value for that button is passed into the program, which then configures and calls the OkDialog window shown in Figure A-8A. If you click the **Cancel** button instead, that passed parameter value will call the OkDialog window shown in Figure A-8B.

Figure A-8
OkDialog windows

A. **OK** in Figure A-7 clicked

B. **Cancel** in Figure A-7 clicked

Winulib user-library reference

AbortRetryIgnoreDialog user module

Overview

This dialog box provides you with the **Abort**, **Retry**, or **Ignore** decision prompts. As shown in Figure A-9, up to four lines of text can be placed in the window.

Figure A-9
AbortRetryIgnoreDialog

Formulator
User Libraries: Winulib
User Modules: AbortRetryIgnoreDialog

	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

AbortRetryIgnoreDialog displays a dialog box containing the passed message strings and has three buttons: **Abort**, **Retry**, and **Ignore**.

Syntax:

status = AbortRetryIgnoreDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char *Message3Text, char *Message4Text);

INPUTS:

NumberOfMessages	(int) The number of 40-character text lines to display.
Message1Text	(char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.
Message2Text	(char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.
Message3Text	(char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.
Message4Text	(char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

3	The **Abort** button was pressed.
4	The **Retry** button was pressed.
5	The **Ignore** button was pressed.
-**10050**	(WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.
-**10051**	(WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.
-**10052**	(WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

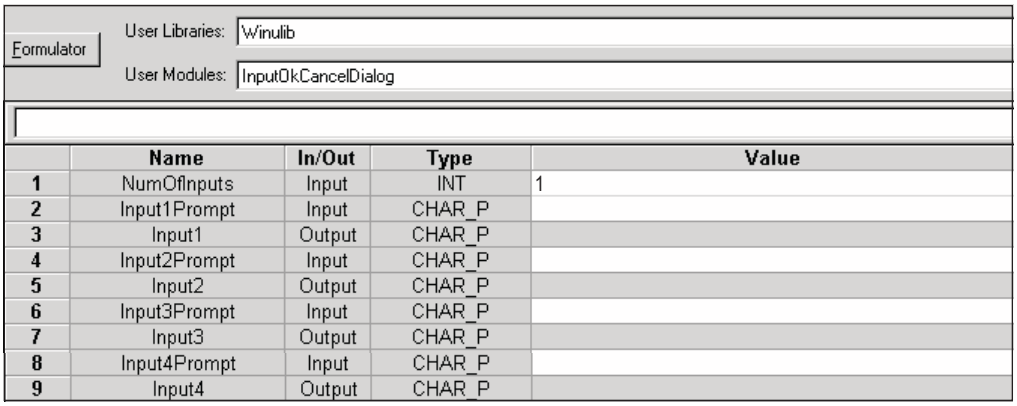
```
status = AbortRetryIgnoreDialog(1, "This is a one line message", "", "", "");
status = AbortRetryIgnoreDialog(4, "Line one", "Line two", "Line three", "Line four");
```

InputOkCancelDialog user module

Overview

This input dialog box allows you to prompt for up to four input parameters. As shown in Figure A-10, there is a separate user-entered prompt message for each input.

Figure A-10
InputOkCancelDialog



	Name	In/Out	Type	Value
1	NumOfInputs	Input	INT	1
2	Input1Prompt	Input	CHAR_P	
3	Input1	Output	CHAR_P	
4	Input2Prompt	Input	CHAR_P	
5	Input2	Output	CHAR_P	
6	Input3Prompt	Input	CHAR_P	
7	Input3	Output	CHAR_P	
8	Input4Prompt	Input	CHAR_P	
9	Input4	Output	CHAR_P	

User-module description

InputOkCancelDialog displays a dialog box containing up to four message prompts and four text input fields with **OK** and **Cancel** buttons.

Syntax:

status = InputOkCancelDialog(int NumOfInputs, char *Input1Prompt, char *Input1, char *Input2Prompt, char *Input2, char *Input3Prompt, char *Input3, char *Input4Prompt, char *Input4);

INPUTS:

NumOfInputs (int) The number of 40-character text lines to display.

Input1Prompt (char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.

Input2Prompt (char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.

Input3Prompt (char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.

Input4Prompt (char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

Input1 (char *) A character buffer for the first user input field. Any text that the user inputs in the first displayed field will be stored here.

Input2 (char *) A character buffer for the second user input field. Any text that the user inputs in the second displayed field will be stored here.

Input3 (char *) A character buffer for the third user input field. Any text that the user inputs in the third displayed field will be stored here.

Input4 (char *) A character buffer for the fourth user input field. Any text that the user inputs in the fourth displayed field will be stored here.

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

1	The **OK** button was pressed.
2	The **Cancel** button was pressed.
-**10050**	(WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.
-**10051**	(WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.
-**10052**	(WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

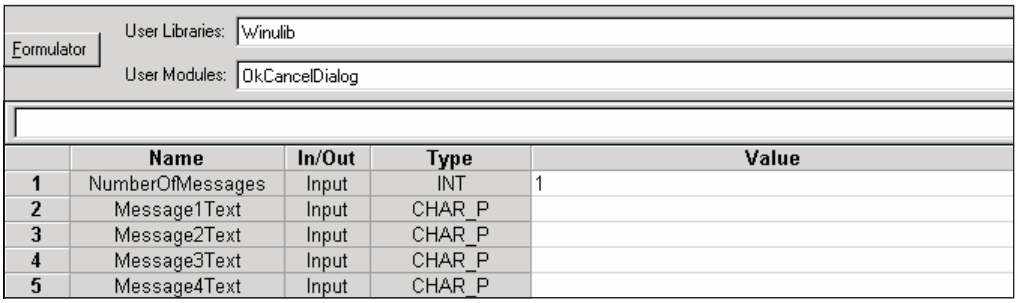
```
status = InputOkCancelDialog(1, "This is a one line message", text1, "", text2, "", text3, "", text4);
status = InputOkCancelDialog(4, "Line one", text1, "Line two", text2, "Line three", text3, "Line four",
        text4);
```

OkCancelDialog user module

Overview

This dialog box provides you with the **OK** or **Cancel** decisions. As shown in Figure A-11, up to four lines of text can be placed in the window.

Figure A-11
OkCancelDialog



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

OkCancelDialog displays a dialog box containing up to four text messages with **OK** and **Cancel** buttons.

Syntax:

```
status = OkCancelDialog(int NumberOfMessages, char *Message1Text, char *Message2Text,
        char *Message3Text, char *Message4Text);
```

INPUTS:

NumberOfMessages	(int) The number of 40-character text lines to display.
Message1Text	(char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.
Message2Text	(char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.
Message3Text	(char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.
Message4Text	(char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

1	The **OK** button was pressed.
2	The **Cancel** button was pressed.
-**10050**	(WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.
-**10051**	(WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.
-**10052**	(WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

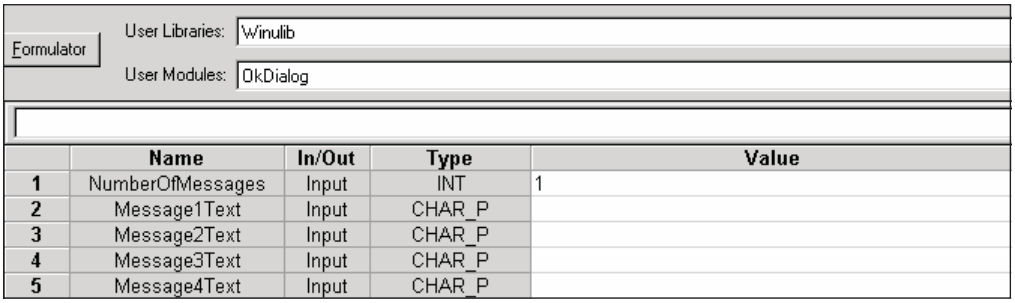
```
status = OkCancelDialog(1, "This is a one line message","", "", "");
status = OkCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

OkDialog user module

Overview

This dialog box is used to pause the test sequence to make an announcement (for example, test finished) or prompt for an action (for example, connection change). Clicking **OK** continues the test sequence. As shown in Figure A-12, up to four lines of text can be placed in the window.

Figure A-12
OkDialog



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

OkDialog displays a dialog box containing up to four lines of text and an **OK** button.

Syntax:

status = OkDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char *Message3Text, char *Message4Text);

INPUTS:

NumberOfMessages	(int) The number of 40-character text lines to display.
Message1Text	(char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.
Message2Text	(char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.
Message3Text	(char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.
Message4Text	(char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

1	The **OK** button was pressed.
-**10050**	(WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.
-**10051**	(WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.
-**10052**	(WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

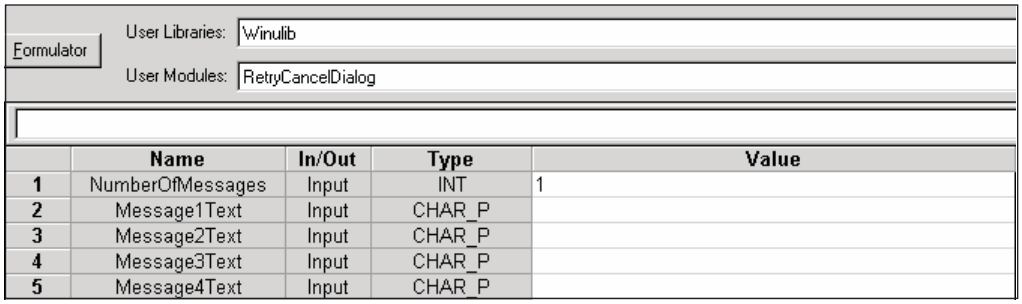
```
status = OkDialog(1, "This is a one line message", "", "", "");
status = OkDialog(4, "Line one", "Line two", "Line three", "Line four");
```

RetryCancelDialog user module

Overview

This dialog box provides you with the **Retry** or **Cancel** decisions. As shown in Figure A-13, up to four lines of text can be placed in the window.

Figure A-13
RetryCancelDialog



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

RetryCanceDialog displays a dialog box containing up to four lines of text as well as **Retry** and **Cancel** buttons.

Syntax:

status = RetryCancelDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char *Message3Text, char *Message4Text);

INPUTS:

NumberOfMessages (int) The number of 40-character text lines to display.

Message1Text (char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.

Message2Text (char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.

Message3Text (char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.

Message4Text (char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

2 The **Cancel** button was pressed.

4 The **Retry** button was pressed.

-**10050** (WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.

-**10051** (WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.

-**10052** (WINULIB_NO_WINDOW_HANDLE) No window handle for *KITE* was found. *KITE* is not running.

Example call:

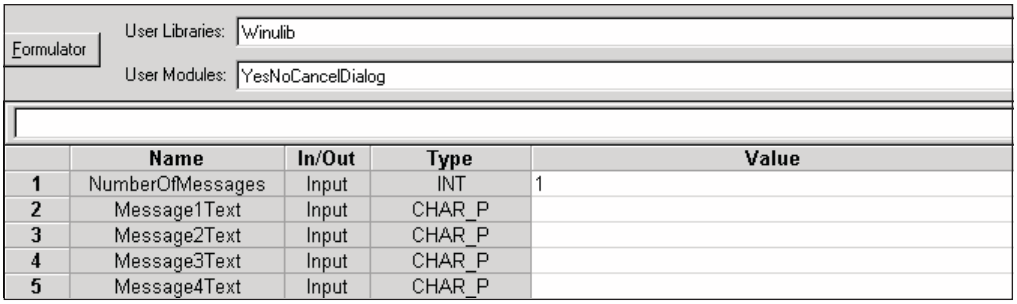
```
status = RetryCancelDialog(1, "This is a one line message", "", "", "");
status = RetryCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

YesNoCancelDialog user module

Overview

This dialog box provides you with the **Yes**, **No**, or **Cancel** decisions. As shown in Figure A-14, up to four lines of text can be placed in the window.

Figure A-14
YesNoCancelDialog



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

YesNoCancelDialog displays a dialog box containing up to four lines of text and **Yes**, **No**, or **Cancel** buttons.

Syntax:

status = YesNoCancelDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char *Message3Text, char *Message4Text);

INPUTS:

NumberOfMessages (int) The number of 40-character text lines to display.

Message1Text (char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.

Message2Text (char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.

Message3Text (char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.

Message4Text (char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

2 The **Cancel** button was pressed.

6 The **Yes** button was pressed.

7 The **No** button was pressed.

-**10050** (WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.

-**10051** (WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.

-**10052** (WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

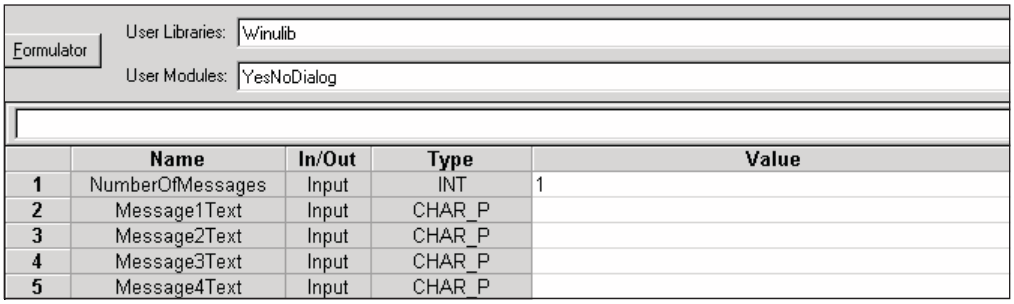
```
status = YesNoCancelDialog(1, "This is a one line message", "", "", "");

status = YesNoCancelDialog(4, "Line one", "Line two", "Line three", "Line four");
```

YesNoDialog user module

Overview

This dialog box provides you with the **Yes** or **No** decisions. As shown in Figure A-15, up to four lines of text can be placed in the window.

Figure A-15
YesNoDialog



	Name	In/Out	Type	Value
1	NumberOfMessages	Input	INT	1
2	Message1Text	Input	CHAR_P	
3	Message2Text	Input	CHAR_P	
4	Message3Text	Input	CHAR_P	
5	Message4Text	Input	CHAR_P	

User-module description

YesNoDialog displays a dialog box containing up to four lines of text along with **YES** and **NO** buttons.

Syntax:

```
status = YesNoDialog(int NumberOfMessages, char *Message1Text, char *Message2Text, char
         *Message3Text, char *Message4Text);
```

INPUTS:

NumberOfMessages (int) The number of 40-character text lines to display.

Message1Text (char *) The text to display on the first line of the dialog box. This line must be less than 40 characters.

Message2Text (char *) The text to display on the second line of the dialog box. This line must be less than 40 characters.

Message3Text (char *) The text to display on the third line of the dialog box. This line must be less than 40 characters.

Message4Text (char *) The text to display on the fourth line of the dialog box. This line must be less than 40 characters.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the data spreadsheet (**Sheet** tab).

6	The **Yes** button was pressed.
7	The **No** button was pressed.
-**10050**	(WINULIB_ILLEGAL_NUM_MSG) An illegal number of messages was specified.
-**10051**	(WINULIB_ILLEGAL_STRING_LEN) The length of one or more messages was too long.
-**10052**	(WINULIB_NO_WINDOW_HANDLE) No window handle for KITE was found. KITE is not running.

Example call:

```
status = YesNoDialog(1, "This is a one line message", "", "", "");

status = YesNoDialog(4, "Line one", "Line two", "Line three", "Line four");
```

Appendix B
Using Switch Matrices

In this section:

Key concepts

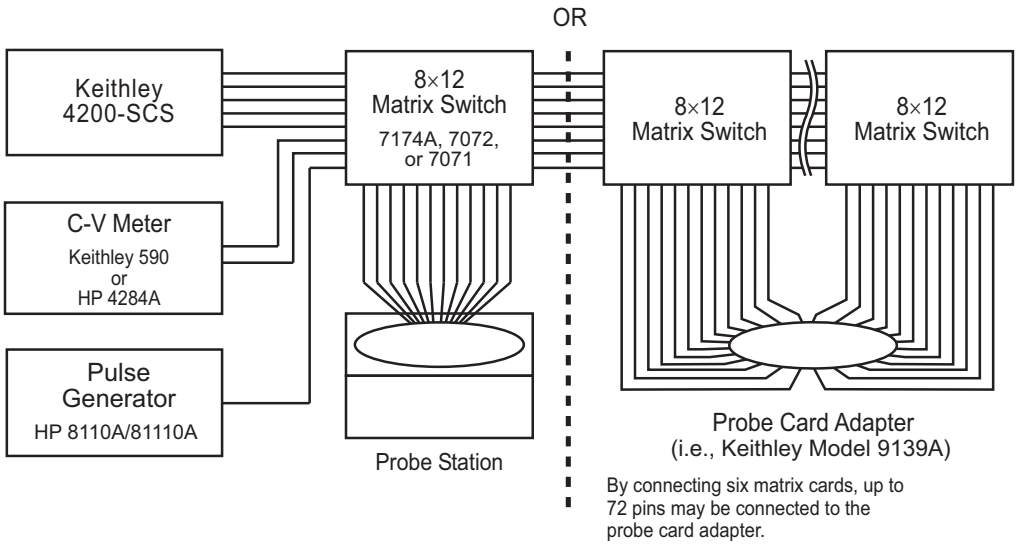
Typical test systems using a switch matrix

A switch matrix provides automatic switching for test instrumentation and devices under test (DUTs). Typical switch matrix systems are shown in Figure B-1.

For low pin-count applications, the system can consist of a single matrix card installed in a Keithley Instruments Model 708A mainframe to provide 12 pins of matrix switching. For higher pin-count applications, six matrix cards can be installed in a Model 707A to provide up to 72 pins of switching.

Figure B-1 shows switch matrix cards connected to a probe station in order to test a wafer. However, a probe station could be replaced by a test fixture to test discrete devices.

Figure B-1
Typical systems using a switch matrix



Matrix card types

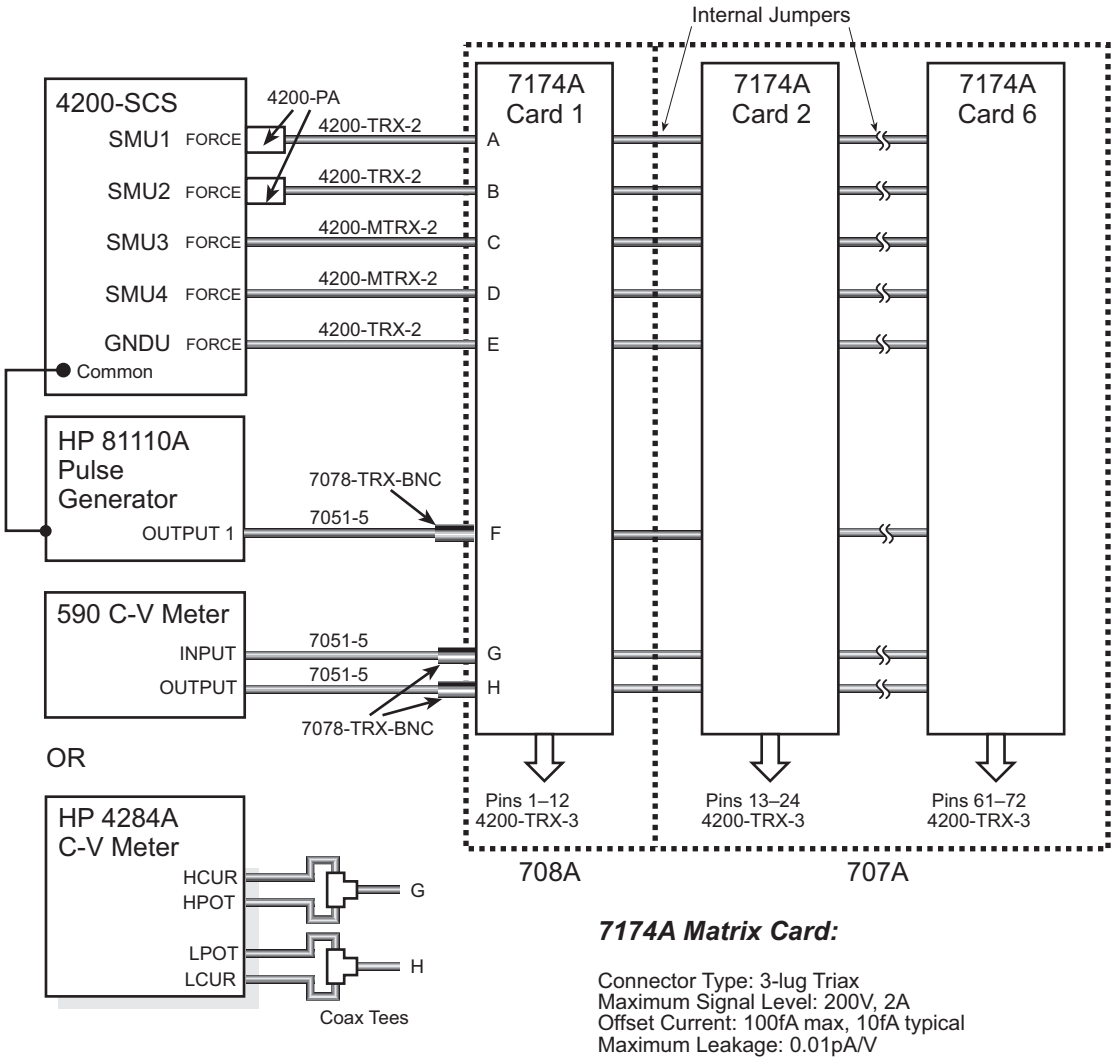
Model 7174A Low Current Matrix Card

The Model 7174A provides high quality, high performance switching of I-V and C-V signals. This matrix card uses 3-pole switching (HI, LO, Guard) with 10fA typical offset current. The card is equipped with 3-lug triax connectors for signal connections.

Figure B-2 and Figure B-3 show test systems using Model 7174A matrix cards. The supplied triax cables connect the Model 4200-SCS directly to matrix rows. The other instruments in the system are fitted with BNC connectors requiring the use of BNC-to-triax adapters.

Local sensing: The system in Figure B-2 uses local sensing. Note that coaxial tees are used to adapt the HP 4980A LCR meter for two-terminal operation.

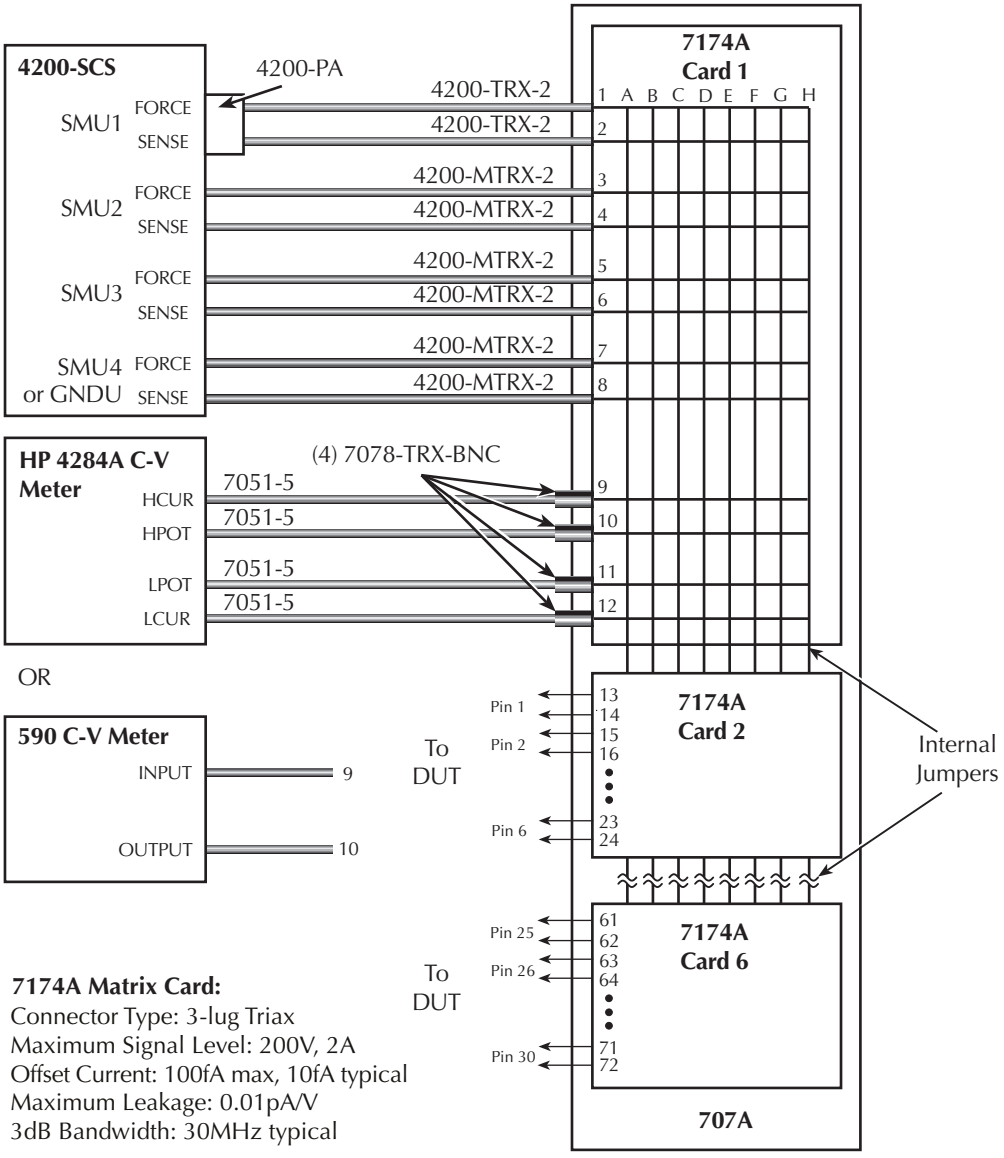
Figure B-2
Test system using Model 7174A matrix cards



Remote sensing: Figure B-3 shows how to connect instrumentation for remote sense operation. Since there are not enough matrix rows, the instruments are connected to the matrix columns. In this configuration, two switch relays are closed to complete a path from an instrument to a DUT. With five DUT matrix cards installed in a Model 707A mainframe, up to 30 DUT pin-pairs can be used.

NOTE: *Figure B-3, Figure B-12, and Figure B-13 show how signals are routed through Model 7174A matrix switches to a DUT.*

Figure B-3
Remote sense test system using Model 7174A matrix cards



NOTE: For this example, instrumentation is connected to matrix columns. Therefore, the switch matrix is rotated 90° for illustration purposes.

Model 7072 Semiconductor Matrix Card

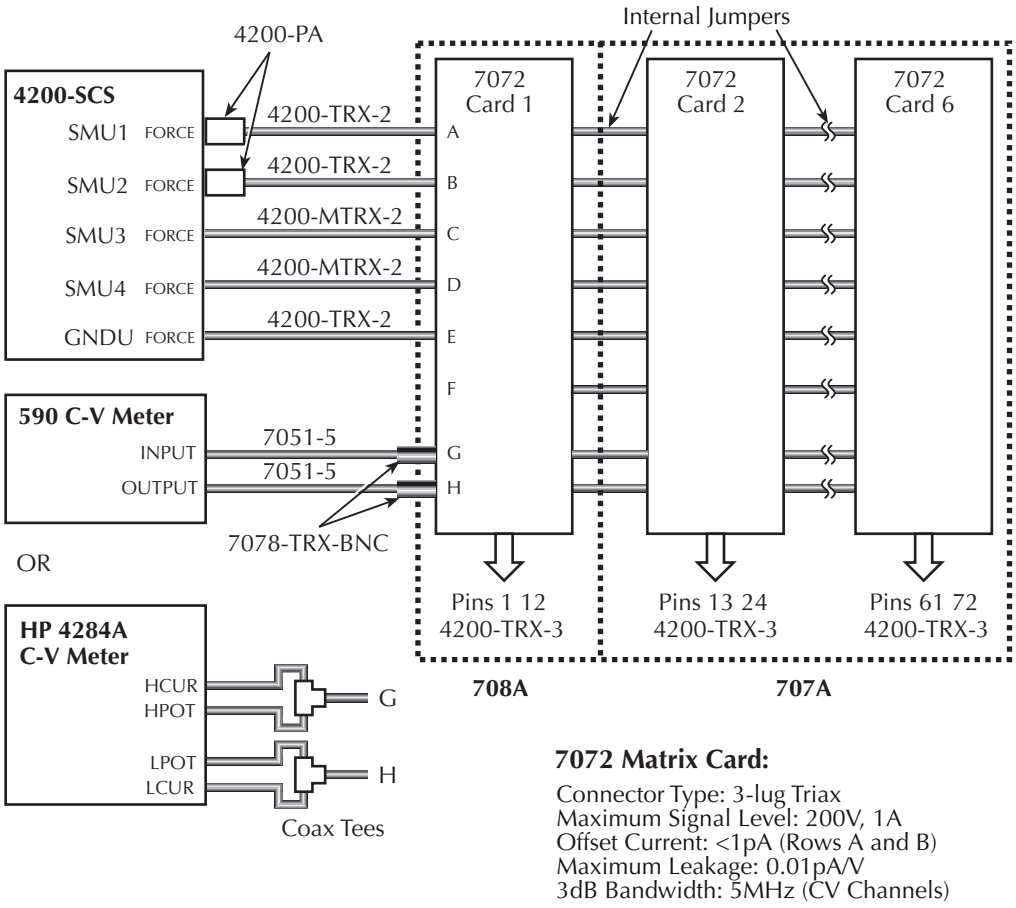
The Model 7072 provides two two-pole low current paths that have <1pA offset current (rows A and B), two 1-pole CV paths for characterization from DC to 1 MHz (rows G and H), and four two-pole paths for general purpose switching (rows C, D, E, and F). The card is equipped with 3-lug triax connectors for signal connections.

Figure B-4 shows a test system using Model 7072 matrix cards. The connection requirements for this card are the same as the connection requirements for the Model 7174A. Notice that the C-V meter is connected to rows G and H. These two rows are optimized for C-V measurements.

If using preamps with the Model 4200-SCS, they should be connected to the first two rows of the Model 7072 matrix card.

NOTE: *Figure B-4 and Figure B-10 through Figure B-12 show how signals are routed through Model 7072 matrix switches to a DUT.*

Figure B-4
Test system using Model 7072 matrix cards



Model 7071 General Purpose Matrix Card

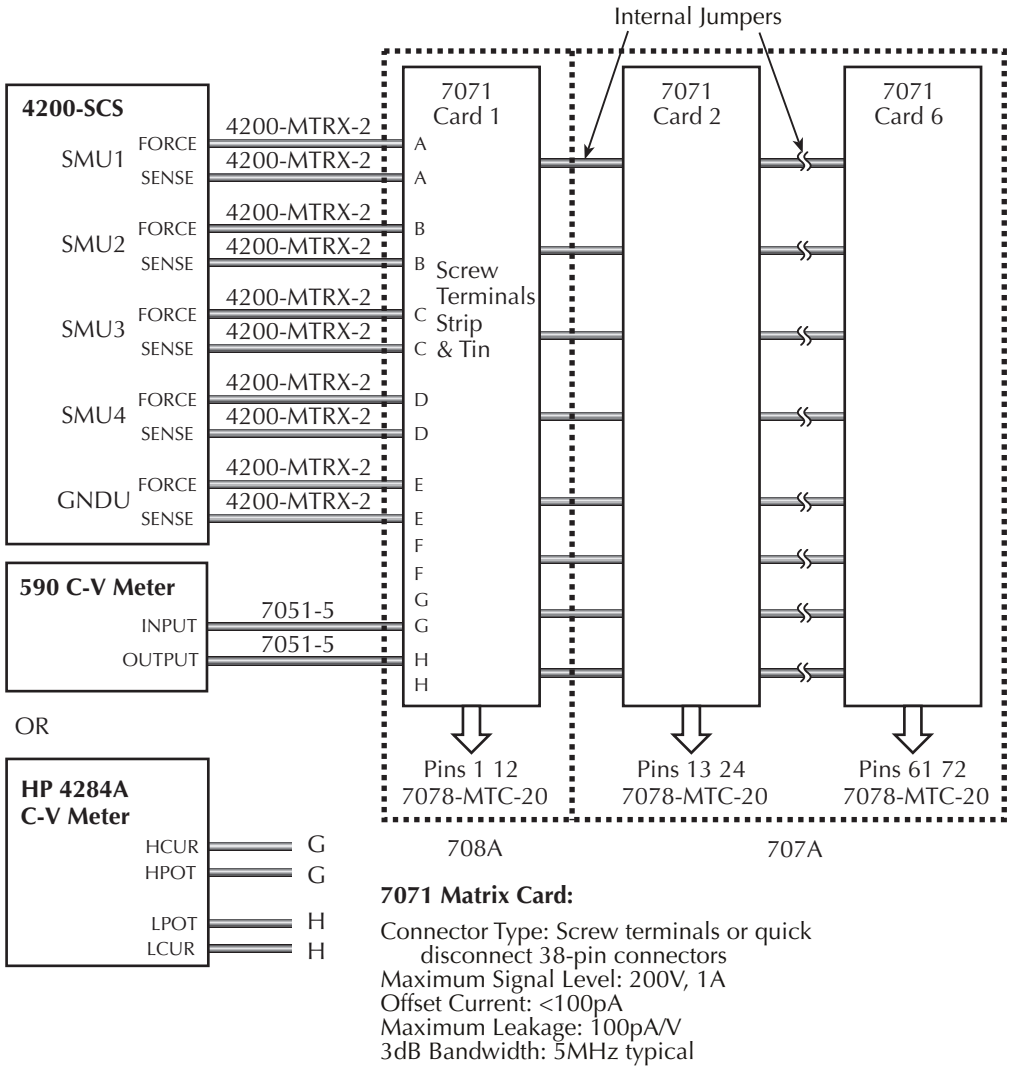
The Model 7071 provides cost-effective switching of I-V and C-V signals. This matrix card uses 3-pole switching (high, low, and guard) with <100 pA offset current. The card is equipped with screw terminals and 38-pin connectors for signal connections.

Figure B-5 shows a test system using Model 7071 matrix cards. The triax and BNC cables are unterminated on one end to allow direct hard-wire connections to the screw terminals of the matrix card.

The test system in Figure B-5 uses remote sensing. Notice that each FORCE and SENSE terminal-pair of the Model 4200-SCS shares the same path (row).

NOTE: *Figure B-5 shows how signals are routed through Model 7071 matrix switches to a DUT.*

Figure B-5
Test system using Model 7071 matrix cards



7071 Matrix Card:

Connector Type: Screw terminals or quick disconnect 38-pin connectors
Maximum Signal Level: 200V, 1A
Offset Current: <100pA
Maximum Leakage: 100pA/V
3dB Bandwidth: 5MHz typical

Switch matrix mainframes

The Model 707A switching mainframe has six slots for matrix cards, while the Model 708A has one slot.

Card installation

To install a matrix card, line the card up with the card guides in the slot and slide it into the slot until it fully seats with the backplane connector. Finger-tighten the spring-loaded mounting screws. Note that if using the screw terminals of the Model 7071, you will have to wire the card before installing it. For details on installation refer to the Models 707A and 708A Instruction Manual.

GPIB connections

The Model 4200-SCS controls the switch matrix via the GPIB. Connect the GPIB port of the mainframe to the Model 4200-SCS using a Model 7007-1 or 7007-2 GPIB cable.

Matrix expansion

When using the Model 707A mainframe, the rows of installed matrix cards can be connected together to increase the number of matrix columns. With six cards installed in the Model 707A, the number of matrix columns can be increased to 72.

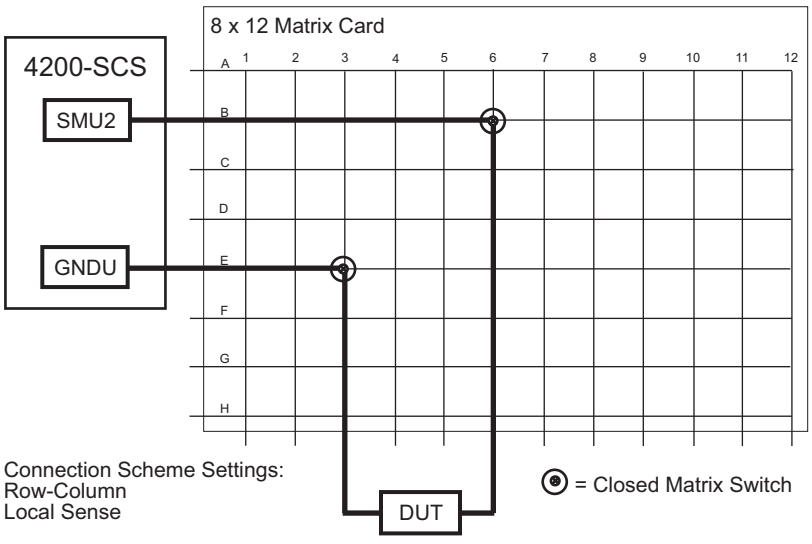
- **Model 7071 matrix card**: The matrix rows of the installed cards are automatically connected through the backplane of the mainframe. No additional connections are required.
- **Model 7072 matrix card**: Rows C through F of the installed cards are automatically connected through the backplane of the mainframe. Internal jumpers must be used for rows A, B, G, and H. See the Model 7072 Instruction Manual for details.
- **Model 7174A matrix card**: Internal jumpers must be used to connect the rows of the matrix cards installed in the mainframe. See the Model 7174A Instruction Manual for details.

Switch matrix control

The **connect** user test module (UTM) utilizes the **ConnectPins** user module to control a switch matrix. The user simply specifies instrument terminal / pin pairs. For example, for the row-column connection scheme shown in Figure B-6, the user-entered parameters **SMU2, 6** for ConnectPins would connect SMU2 to Pin 6, and **GNDU, 3** would connect GNDU (ground unit) to Pin 3.

A matrix control example using the **ConnectPins** user module is provided in Matrix control example presented later in this appendix. Detailed information for ConnectPins is provided in Matrixulib user library reference later in this appendix.

Figure B-6
Row-column connection scheme



Connection scheme settings

The following connection scheme settings are set from the Keithley CONFiguration (KCON) utility when the switch matrix is added to the system configuration. See Using KCON to add a switch matrix to the system later in this appendix.

Row-column / instrument card settings

Row-column: For the row-column setting, instruments are connected to the matrix rows, and the DUT is connected to the matrix columns (see Figure B-6). However, if the instrumentation requirements exceed eight paths (rows), you will have to use the instrument card configuration.

Instrument Card: For the Instrument Card setting, both the instrumentation and the DUT are connected to matrix columns (see Figure B-8). No external connections are to be made to matrix rows. In this configuration, two switch relays are closed to complete a path from an instrument to a DUT.

Local Sense / Remote Sense settings

Local Sense: With Local Sense selected, only the connection paths specified by the connect UTM are completed. For example, in Figure B-6, the specified connection paths would be **SMU2, 6** (connect SMU2 to Pin 6), and **GNDU, 3** (connect GNDU to Pin 3).

Remote Sense: With Remote Sense selected, rows and columns are paired together as follows:

Row A paired with row B	Column 1 paired with Column 2
Row C paired with row D	Column 3 paired with Column 4
Row E paired with row F	Column 5 paired with Column 6
Row G paired with row H	Column 7 paired with Column 8
	Column 9 paired with Column 10
	Column 11 paired with Column 12

When you specify a connection path in the connect UTM, the paired connection path will also be completed. For example, in Figure B-8, the specified connection paths would be **SMU1, 4** (connect SMU1 to Pin 4) and **GNDU, 3** (connect GNDU to Pin 3).

Signal paths to a DUT

Figures B-1 through B-13 show signal path examples from the various test instruments through the matrix switches to a DUT.

Model 4200-SCS signal paths

Figure B-7 shows remote sensing (4-wire) signal paths through a matrix card using two-pole switching. Two-pole switching is provided by the Models 7174A and 7072 (rows A through F).

Sense setting: Remote sensing must be selected in order to make the connections shown in Figure B-7. With remote sensing selected, rows and columns are paired together as follows:

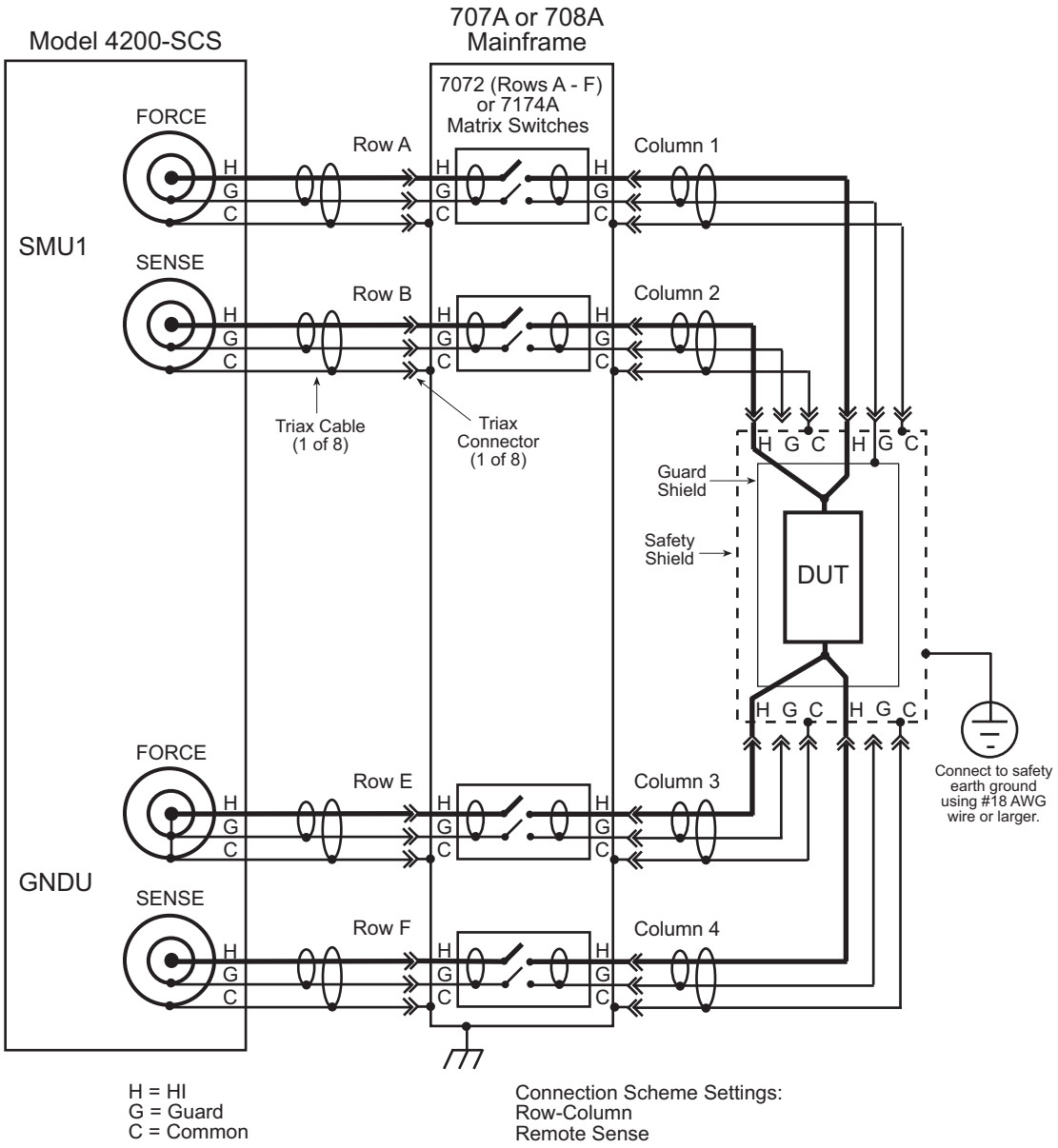
Row A (force) paired with row B (sense)	Column 1 (force) paired with column 2 (sense)
Row E (force) paired with row F (sense)	Column 3 (force) paired with column 4 (sense)

When the FORCE matrix switches are closed by the **ConnectPins** user module, the SENSE matrix switches will also close.

For local sensing (2-wire), the connections from the SENSE terminals of the Model 4200-SCS would not be used.

NOTE: *See Connection scheme settings earlier in this section, for details on local and remote sensing.*

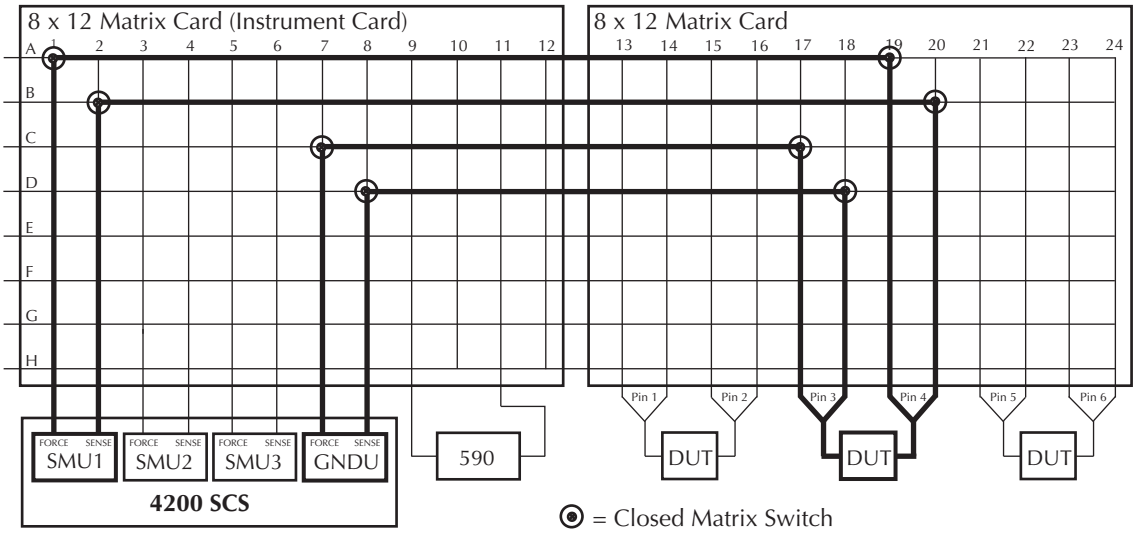
Figure B-7
Model 4200-SCS signal paths through a two-pole matrix card using remote sensing



Connection setting: The row-column setting must be used when connecting instrumentation to matrix rows, as shown in Figure B-7. However, the maximum number of rows available to the test system is eight. Therefore, if instrumentation needs more than eight pathways, then they must instead be connected to matrix columns, and the instrument card setting must be used. Figure B-8 shows a test system with both the instruments and the DUT connected to matrix columns.

NOTE: *See Connection scheme settings earlier in this section, for details on the row-column and Instrument Card settings.*

Figure B-8
Instrument card connection scheme



Connection Scheme Settings:
Instrument Card
Remote Sense

NOTE: The Model 4200-SCS will automatically select the first availabl rows to make connections to the DUT. In this example, Rows A through D are the first available rows.

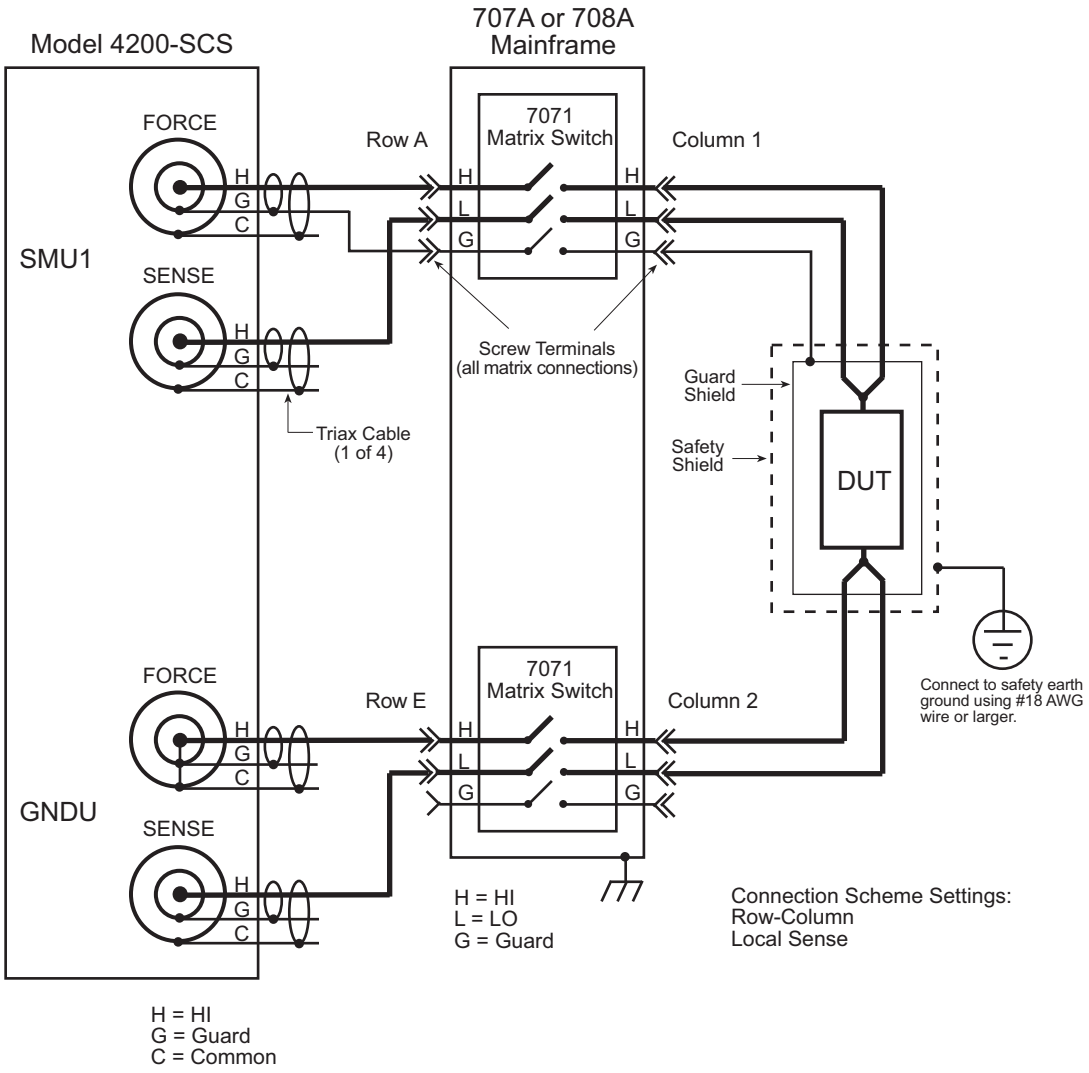
Figure B-9 shows Model 4200-SCS signal paths through a 3-pole Model 7071 matrix card using remote sensing. Note that for this configuration, each FORCE and SENSE connector does not use a separate path (row). Unlike the configuration shown in Figure B-7, each FORCE/SENSE connector pair is routed through a single 3-pole matrix switch. Since row pairing is not required, the Local Sense setting must be used.

For two-wire local sense connections, do not use the SENSE connectors of the Model 4200-SCS.

WARNING ***The guard (G) terminals of a source measure unit (SMU) are at the same potential as SMU HI (H). Therefore, if a hazardous voltage is present on SMU HI, it is also present on SMU Guard. Ensure unconnected guard cables are insulated to prevent electric shock that could cause personal injury or death.***

NOTE: *See Connection scheme settings earlier in this section, for details on sense settings.*

Figure B-9
Model 4200-SCS signal paths through a 3-pole matrix card using remote sensing



CV Analyzer signal paths

Figure B-10 and Figure B-11 show local sense, CV Analyzer signal paths through rows B and H of a Model 7072 matrix card. A CV analyzer can be used with any of the three matrix card types; however, rows G and H of the Model 7072 are optimized for C-V measurements.

Figure B-10
Model 590 signal paths through Model 7072 matrix card using local sensing

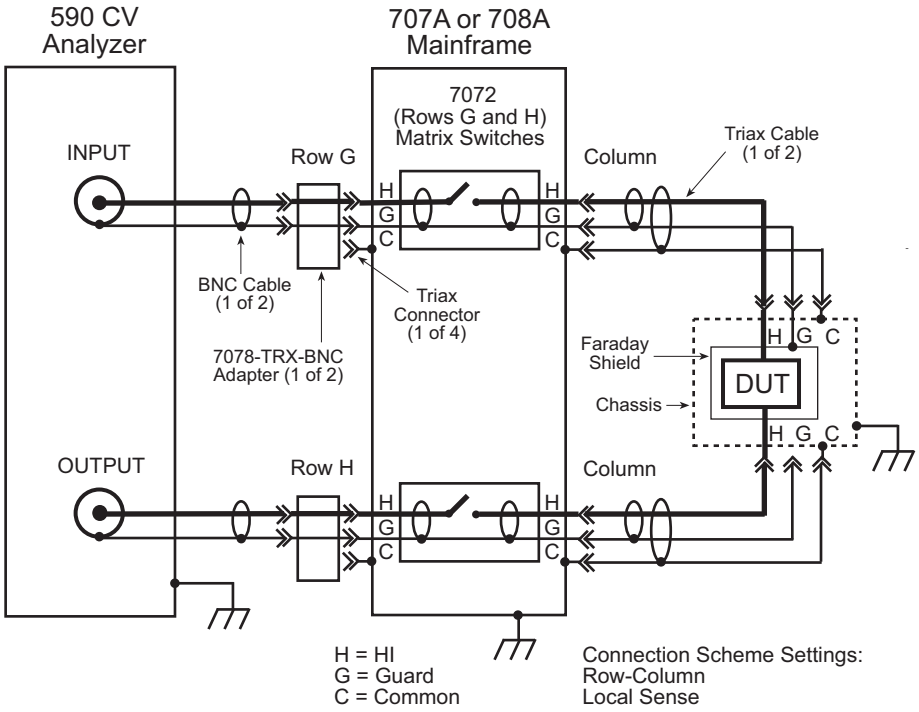


Figure B-11
HP Model 4980A signal paths through Model 7072 matrix card using local sensing

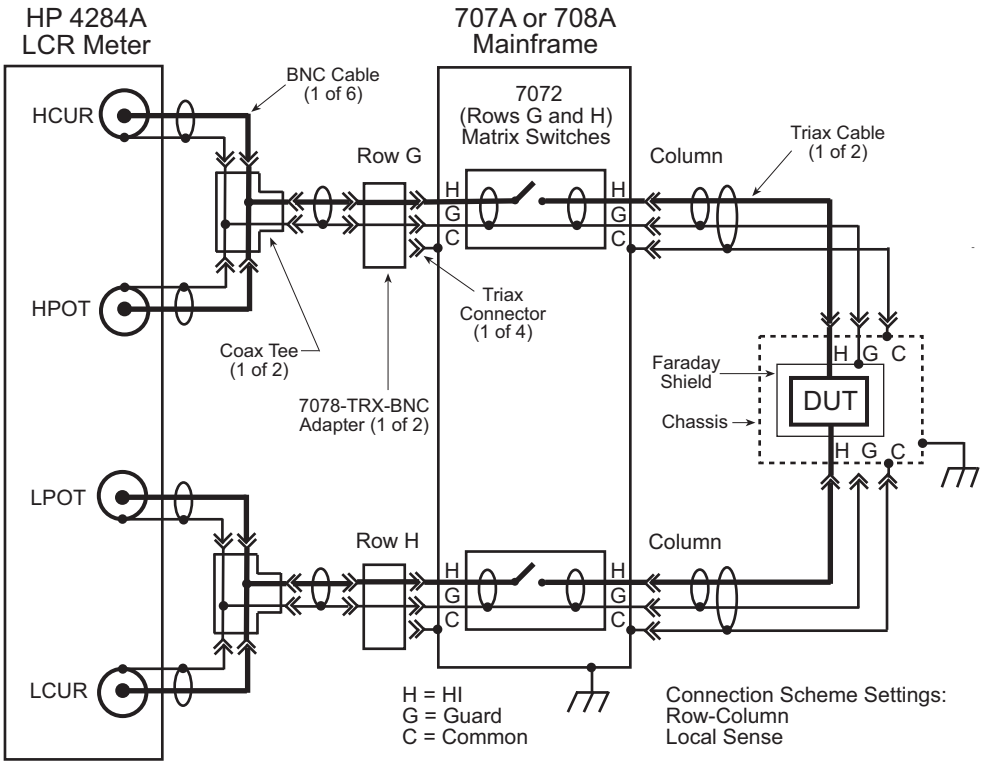
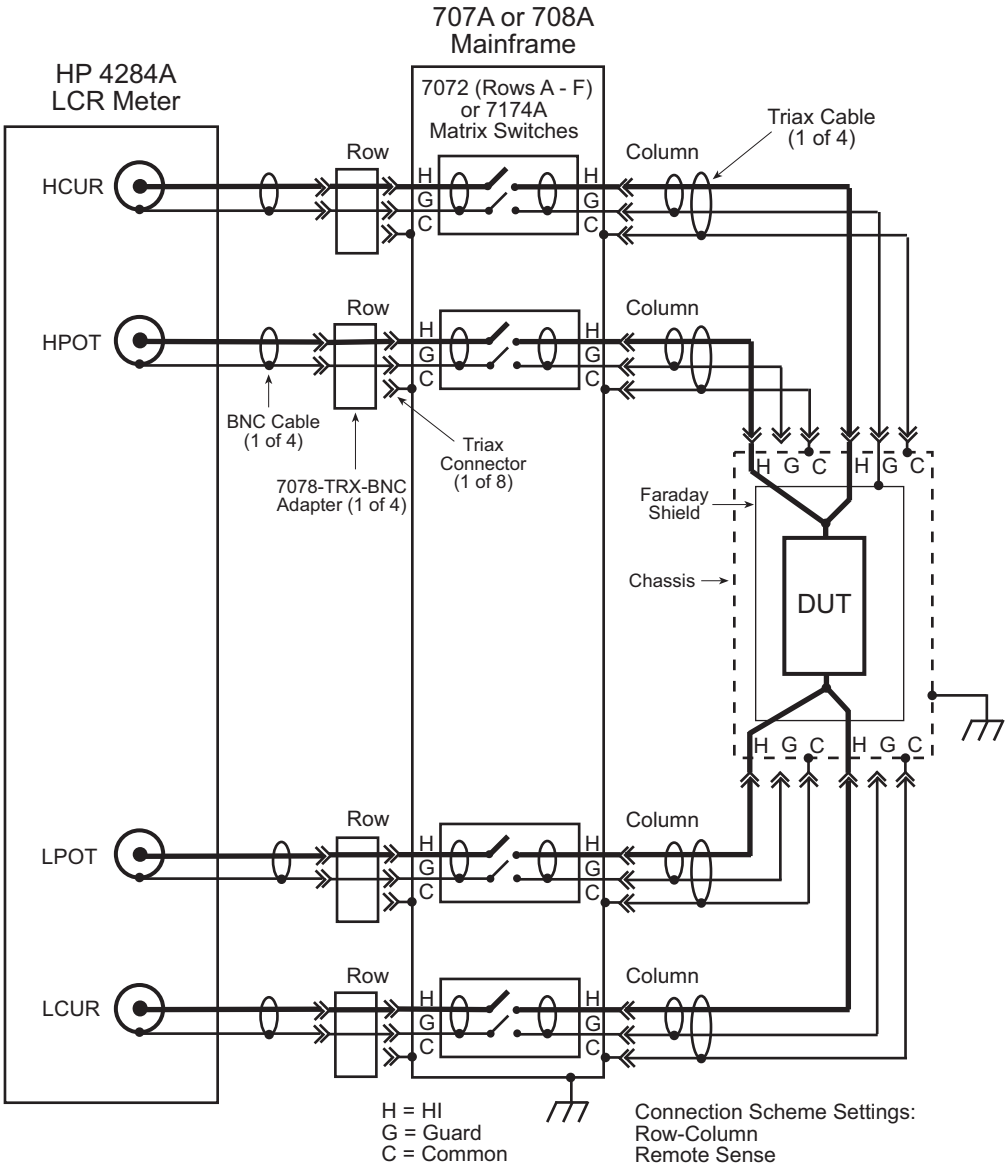


Figure B-12 shows the remote sense signal paths for the HP Model 4980A LCR meter through a 2-pole matrix card. Since row pairing is required, the Remote Sense setting must be used.

Figure B-12
HP Model 4980A signal paths through a two-pole matrix card using remote sensing

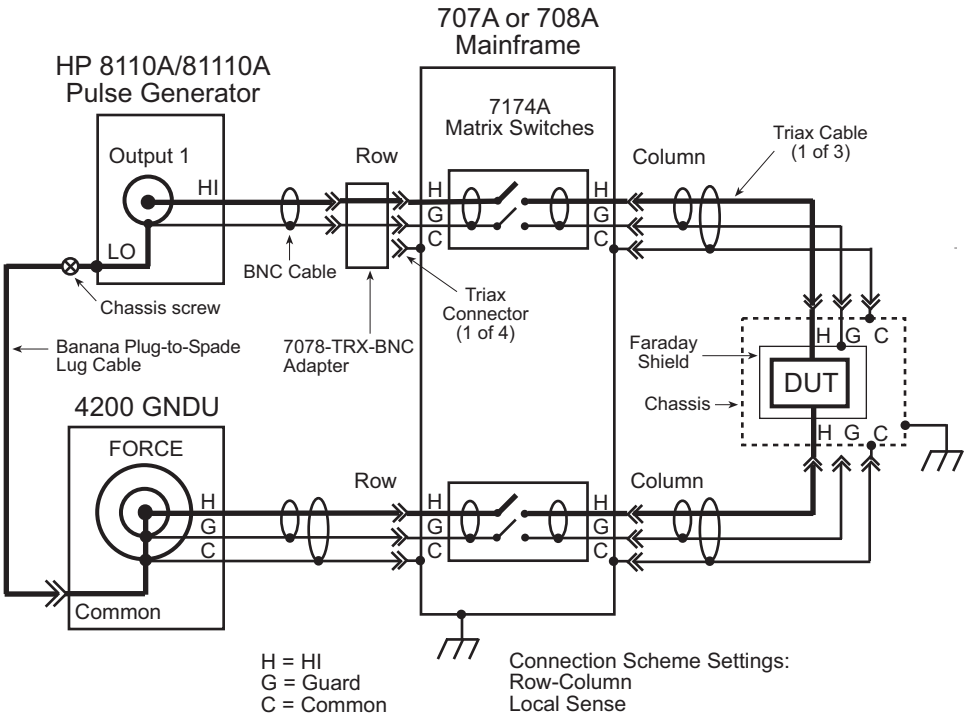


HP Model 8110A/81110A pulse generator signal path

Figure B-13 shows the HI signal path through the Model 7174A matrix card. However, the pulse generator can also be used with other two matrix card types.

Note that the pulse generator LO is not routed through the matrix card. A separate external return path is required. The chassis of the pulse generator is output LO. As shown in Figure B-13, use a banana plug cable that is terminated with a spade lug on one end. Connect the banana plug end of the cable to the Common banana jack of the GNDU, and attach the spade lug end to a chassis screw on the pulse generator.

Figure B-13
HP Model 8110A/81110A signal path through a Model 7174A matrix card



Using KCON to add a switch matrix to the system

In order for the Model 4200-SCS to control a switch matrix, the switch matrix must be added to the system configuration. The switch matrix is added to the test system using the KCON (Keithley CONfiguration utility).

The switch matrix can be used with a test fixture to test discrete DUT, or with a probe station to test a wafer. The test fixture or probe station is also added to the system configuration using KCON.

Step 1. Close KITE and open KCON

Close KITE by clicking the close button (X) at the top right-hand corner of the KITE panel. If you have made changes, you will be prompted to save them. On the windows desktop, double-click the **KCON** icon to open KCON.

For details on using KCON, see Keithley CONfiguration Utility (KCON) in Section 7.

Step 2. Add a test fixture or probe station

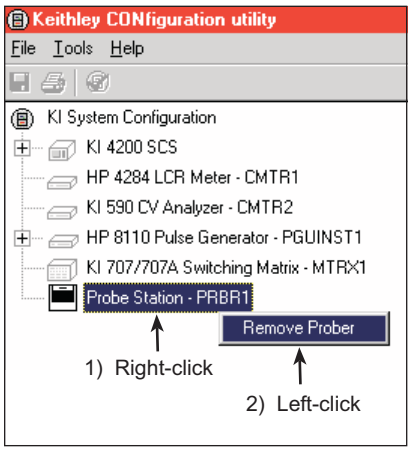
NOTE: *Using a switch matrix requires a test fixture or probe station in the system configuration.*

A test fixture or a probe station can be added to the test system. However, both cannot be in the system configuration together. If a probe station is already added to the system, it will have to be removed in order to add a test fixture. Conversely, if a test fixture is in the system, it will have to be removed before you can add a probe station.

Add a test fixture

If a probe station is in the system configuration it will have to be removed. Figure B-14 shows how to remove it from the system. Right-click **Probe Station PRBR1** to display the **Remove Prober** dialog box, then left-click **Remove Prober.** A dialog box will ask for confirmation. Click **Yes** to remove the prober.

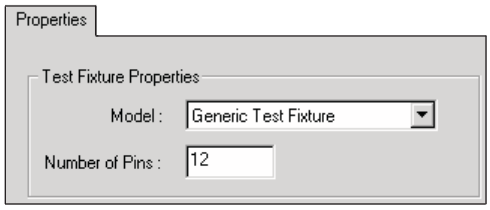
Figure B-14
Removing a component from the system configuration



Perform the following steps to add a test fixture to the system configuration:

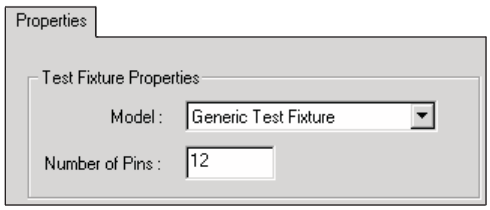
1. As shown in Figure B-15, select **Add External Instrument > Test Fixture** from the **Tools** menu.

Figure B-15
Test fixture properties



2. From the drop-down menu in the **Test Fixture Properties** window (Figure B-16), select a test fixture. The three test fixture options from the menu include:
   - **Generic Test Fixture**: If you select this test fixture, specify the number of DUT terminal pins (2 to 72) equipped on the fixture. Figure B-16 shows the **Generic Test Fixture** selected, and the **Number of Pins** set to **12**.

Figure B-16
Test fixture properties



- **Keithley Instruments Model 8006**: If you select this test fixture, the number of pins will be fixed at 12.
- **Keithley Instruments Model 8007**: If you select this test fixture, the number of pins will be fixed at 72.

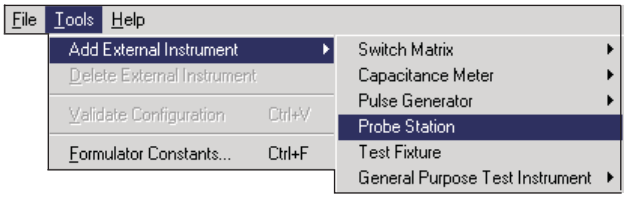
Add a probe station

If a test fixture is already in the system configuration it will have to be removed. Figure B-14 shows how to remove a component from the system. Instead of removing the prober, as shown in Figure B-14, you will be removing the test fixture.

Perform the following steps to add a probe station to the system configuration:

1. As shown in Figure B-17, select **Add External Instrument > Probe Station** using the **Tools** menu.

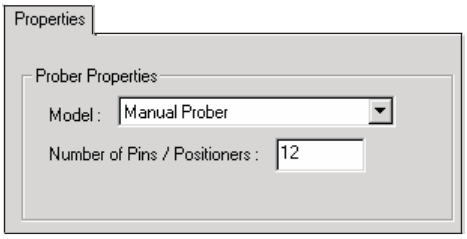
Figure B-17
Tools Menu to add a probe station



2. From the drop-down menu in the **Prober Properties** window (Figure B-18), select a probe station. Supported probe stations include the following:
   - Fake Prober
   - Manual Prober
   - Micromanipulator 8860 Prober
   - Suss MicroTec PA200 Prober

NOTE: *Contact Keithley for the most up-to-date list of supported probers. If using an unsupported prober, you will have to create a user library and module to control it.*

Figure B-18
Prober properties



3. Specify the number of prober terminal pins (2 to 72) equipped on probe station. Figure B-18 shows the **Manual Prober** selected, and the **Number of Pins / Positioners** is set to **12**.

Step 3. Add switching system mainframe

Add the Keithley Instruments Model 707/707A or 708/708 mainframe using the **Tools** menu shown in Figure B-19. Figure B-20 shows the properties setting window for the Model 707/707A. If the Model 708/708A mainframe is selected instead, there will only be one switch card slot.

Figure B-19
Tools menu to add switch matrix

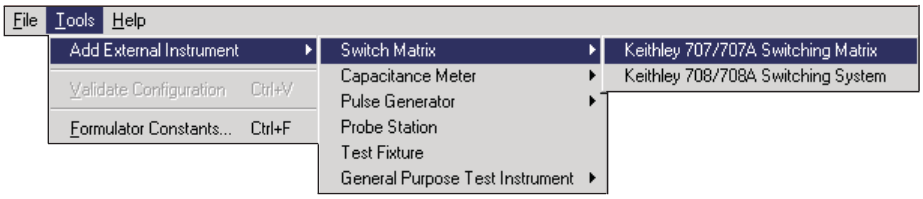
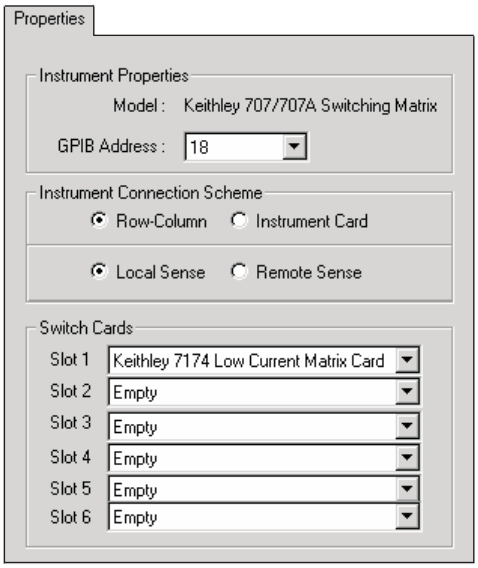


Figure B-20
Model 707/707A properties window



Step 4. Set GPIB address

The GPIB address setting in the **Properties** window (Figure B-20) must match the actual GPIB address of the mainframe. The address for the switch system mainframe is briefly displayed during its power-on sequence.

Step 5. Configure the instrument connection scheme

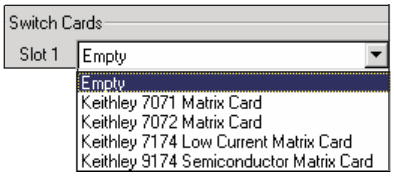
1. **Row-column / instrument card**: Select **row-column** if connecting instrumentation to matrix rows and DUT to matrix columns. Select **instrument card** if all connections (instrumentation and DUT) are to be made to matrix columns only.
2. **Local Sense / Remote Sense**: For two-wire connections to the DUT, select **Local Sense**. For 4-wire connections to the DUT, select **Remote Sense**.

Step 6. Assign matrix card to mainframe slots

Use the pull-down menus (see Figure B-21) to assign slots to each model number of the matrix card (or cards) installed in the mainframe. Set unused slots to **Empty**. Figure B-20 shows that slot 1 is assigned to the Model 7174 matrix card.

NOTE: *You cannot mix matrix card models. For example, if you set slot 1 to Model 7174, all other slots can only be set to Model 7174 (or Empty). To select a different model, you will first have to set slot 1 to Empty.*

Figure B-21
Matrix card models



Step 7. Set matrix card properties

The system configuration is shown on the left side of the KCON window. Figure B-22 shows the Model 7174 Matrix Card installed in slot 1. Clicking **KI 7174 Matrix Card - CARD1** opens the matrix card Properties window for that card (Figure B-23).

Each row and column has a drop-down menu to set the card properties. If the row-column connection scheme is selected, instruments are assigned to the rows, while the test fixture pins or probe pins are assigned to the columns. If the instrument card connection scheme is selected, both instrumentation and test fixture/probe pins are assigned to columns (row assignment boxes are disabled).

Figure B-22
Open card properties window

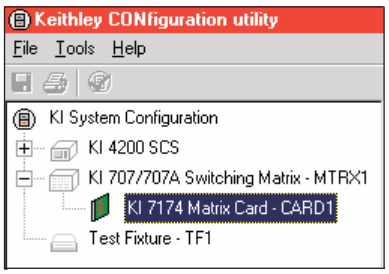
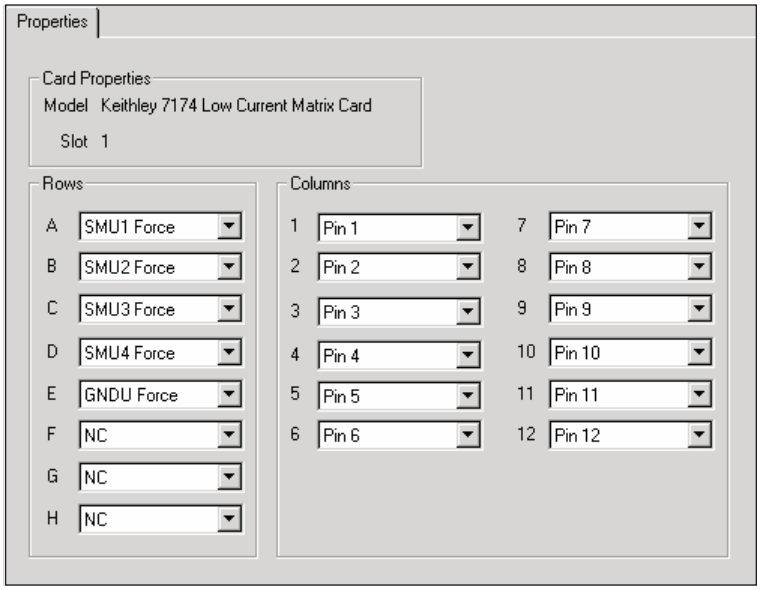


Figure B-23 shows an example configuration for the row-column connection scheme. Note that card properties must match the actual physical connections to the matrix card.

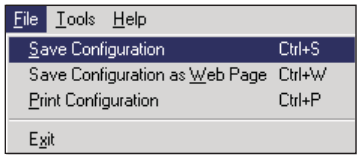
Figure B-23
Card properties window



Step 8. Save configuration

The KCON configuration is saved from the **File** menu on the toolbar. As shown in Figure B-24, click **Save Configuration**.

Figure B-24
Save KCON system configuration



Step 9. Close KCON and open KITE

KCON can be closed from the **File** menu by clicking **Exit**. It can also be closed by clicking the close button (X) at the top right-hand corner of the KCON panel.

On the windows desktop, double-click the **KITE** icon to open KITE.

Switch matrix control example

This example demonstrates how the **ConnectPins** user module controls a switch matrix. You will modify a **connect** UTM to connect SMU2 to a DUT, as shown in Figure B-6. It assumes that the switch matrix is set for row-column connections with Local Sense selected. It also assumes that the matrix card properties are set as shown in Figure B-23.

Detailed information on the **ConnectPins** user module is provided in Matrixulib user library reference later in this appendix. This information is presented after this example.

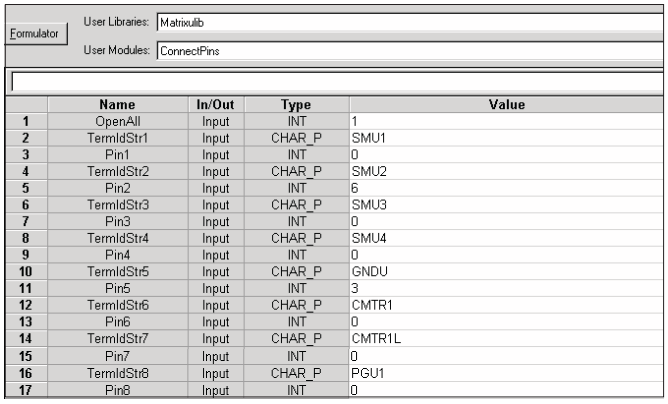
Step 1. Open "connect" UTM

From any KITE project that uses a switch matrix (for example, **ivswitch** project), open a **connect** UTM by double-clicking it in the Project Navigator. The **connect** UTM uses the **ConnectPins** user module.

Step 2. Modify "connect" UTM

Change the **connect** UTM to the parameters shown in Figure B-25. In line 1, parameter value 1 opens all matrix card switches. Lines 4 and 5 connect SMU2 to Pin 6, and lines 10 and 11 connect GNDU to Pin 3. The parameter value 0 for all other terminal/pin pairs indicates no connection will be made.

Figure B-25
Modify connect UTM



	Name	In/Out	Type	Value
1	OpenAll	Input	INT	1
2	TermIdStr1	Input	CHAR_P	SMU1
3	Pin1	Input	INT	0
4	TermIdStr2	Input	CHAR_P	SMU2
5	Pin2	Input	INT	6
6	TermIdStr3	Input	CHAR_P	SMU3
7	Pin3	Input	INT	0
8	TermIdStr4	Input	CHAR_P	SMU4
9	Pin4	Input	INT	0
10	TermIdStr5	Input	CHAR_P	GNDU
11	Pin5	Input	INT	3
12	TermIdStr6	Input	CHAR_P	CMTR1
13	Pin6	Input	INT	0
14	TermIdStr7	Input	CHAR_P	CMTR1L
15	Pin7	Input	INT	0
16	TermIdStr8	Input	CHAR_P	PGU1
17	Pin8	Input	INT	0

Step 3. Run connect UTM

When the **connect** UTM is run by clicking the green **Run** button, the Model 4200-SCS will connect to the DUT as shown in Figure B-6.

Matrixulib user library reference

ConnectPins user module

Overview

This user module allows you to control a switch matrix. The default input parameters are shown in Figure B-26. Typically, OpenAll (line 1) is set to 1 to initially open all connections. If set to 0, the present connections are not affected.

The rest of the input parameters are structured as terminal / pin pairs. Each terminal / pin pair specifies the signal path through the matrix. For example, if the specified pin parameter for SMU1 is 4, then SMU4 will connect to pin 4 of the test fixture or prober when the UTM is run. The pin parameter value 0 (or -1) indicates that no connection will be made.

Terminal ID: Terminal identification for the most common components used in the system configuration are as follows:

- **SMU1 - SMU4**: These are the signal HI terminals for the four SMUs.
- **GNDU**: This is common terminal for the Ground Unit of the Model 4200-SCS.
- **CMTR1**: This is used for a CV Analyzer. For the Model 590, it is the OUTPUT terminal. For the HP Model 4980A, it is the HCUR terminal.

- **CMTR1L**: This is also used for a CV Analyzer. For the Model 590, it is the INPUT terminal. For the HP Model 4980A, it is the LCUR terminal.
- **PGU1**: This is output HI for the HP Model 8110A/81110A Pulse Generator.

NOTE: *A test example demonstrates how this user-module controls the switch matrix (see Switch matrix control example earlier in this appendix).*

Figure B-26
ConnectPins (default parameters)

Formulator | User Libraries: Matrixulib | User Modules: ConnectPins | 0

	Name	In/Out	Type	Value
1	OpenAll	Input	INT	1
2	TermIdStr1	Input	CHAR_P	SMU1
3	Pin1	Input	INT	0
4	TermIdStr2	Input	CHAR_P	SMU2
5	Pin2	Input	INT	0
6	TermIdStr3	Input	CHAR_P	SMU3
7	Pin3	Input	INT	0
8	TermIdStr4	Input	CHAR_P	SMU4
9	Pin4	Input	INT	0
10	TermIdStr5	Input	CHAR_P	GNDU
11	Pin5	Input	INT	0
12	TermIdStr6	Input	CHAR_P	CMTR1
13	Pin6	Input	INT	0
14	TermIdStr7	Input	CHAR_P	CMTR1L
15	Pin7	Input	INT	0
16	TermIdStr8	Input	CHAR_P	PGU1
17	Pin8	Input	INT	0

User module description

The **ConnectPins** module allows you to control your switch matrix. You can connect the instrument terminals to one or more DUT pins. If the DUT pin number is less than 1, then that connection is ignored (not performed), otherwise the specified instrument is connected to the desired DUT pin. If you wish to connect an instrument to more than one DUT pin, you may specify that instrument terminal again in the parameter list.

If the OpenAll parameter is less than one, then the matrix is NOT cleared before making connections; if OpenAll is 1, then all previous matrix connections are cleared before making the new connections.

Syntax:

status = ConnectPins(int OpenAll, char *TermIdStr1, int Pin1, char *TermIdStr2, int Pin2, char *TermIdStr3, int Pin3, char *TermIdStr4, int Pin4, char *TermIdStr5, int Pin5, char *TermIdStr6, int Pin6, char *TermIdStr7, int Pin7, char *TermIdStr8, int Pin8);

INPUTS:

OpenAll (int) This flag controls whether the switch matrix is first cleared before making any new connections. If this parameter is set to 1, then all previous connections are cleared, otherwise they are left intact. Valid inputs: 0 or 1.

TermIdStr1-TermIdStr8 (char *) Terminal identification string. Refers to an instrument as defined by

TermIdStr8 KCON. Valid inputs (configuration dependent) are: SMUn, CMTRn, CMTRnL, PGUn, GPIn, GPInL, GNDU (where n is a number from 1 through 8).

Pin1 - Pin8 (int) The DUT pin number (configuration dependent) to which the instrument will be attached. If a number less than 1 is specified, then no connection will be made. Valid inputs: -1 to 72.

OUTPUTS:

None

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0 OK.

-**10000** (INVAL_INST_ID) An invalid instrument ID was specified. This generally means that there is no instrument with the specified ID in your configuration.

-**10001** (INVAL_PIN_SPEC) An invalid DUT pin number was specified.

-**10003** (NO_SWITCH_MATRIX) No switch matrix was found.

-**10004** (NO_MATRIX_CARDS) No matrix cards were found.

Example:

To connect SMU1 to pin 7, SMU2 to pin 8, SMU3 to pin 12, SMU4 to pin 1, ground pin 15, connect the pulse generator to pin 13, connect the CMTR to pins 9 and 10, and clear the previous connections:

ConnectPins(1, SMU1, 7, SMU2, 8, SMU3, 12, SMU4, 1, GNDU, 15 PGU1, 13, CMTR1, 9, CMTR1L, 10).

Appendix C

Using a Keithley Instruments Model 590 CV Analyzer

In this section:

Key concepts

NOTE: *For details on all aspects of Model 590 operation, refer to the Model 590 CV Analyzer Instruction Manual.*

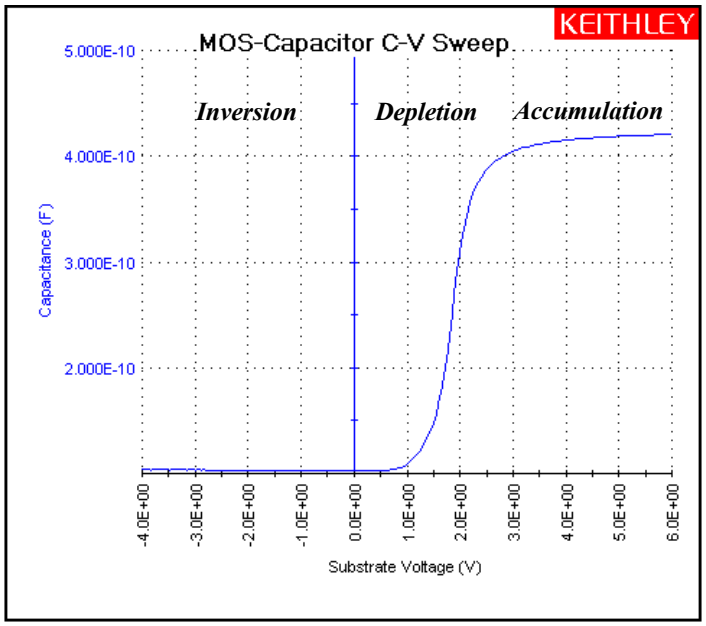
C-V measurement basics

The Keithley Instruments Model 590 CV Analyzer measures capacitance versus voltage (C-V) and capacitance versus time (C-t) of semiconductor devices. Typically, C-V measurements are performed on capacitor-like devices, such as a metal-oxide-silicon capacitor (MOS-C).

The purpose of MOS-C measurements is to study the integrity of the gate oxide and silicon doping profile, and the generation and recombination of silicon semiconductor material. The interface quality between the gate oxide and silicon can also be studied. In addition, other dielectric materials used in an integrated circuit (IC) can be studied using the C-V technique.

The voltage sweeping capability of the Model 590 makes it easy to perform a series of capacitance measurements that span the three regions of a C-V curve: the accumulation region, depletion region, and inversion region. Figure C-1 shows the three regions of a typical C-V curve for a MOS-C.

Figure C-1
Typical C-V curve for a MOS-C



Capacitance measurement tests

The Keithley Instruments Model 4200-SCS provides the following user modules to perform C-V tests using the Model 590:

- **CvSweep590: C-V sweep test**
  Performs a capacitance measurement at each step of a user-configured linear voltage sweep.
- **CvPulseSweep590: C-V pulse sweep test**
  Performs a capacitance measurement at each step of a user-configured pulsed voltage sweep.
- **CtSweep590: C-t measurements**
  Performs a specified number of capacitance measurements at a specified time interval. Voltage is held constant for these capacitance measurements.
- **Cmeas590: C measurement**
  Performs a capacitance measurement at a fixed bias voltage.

NOTE: *Details on all user modules for the Model 590 are provided in this appendix in ki590ulib user library reference.*

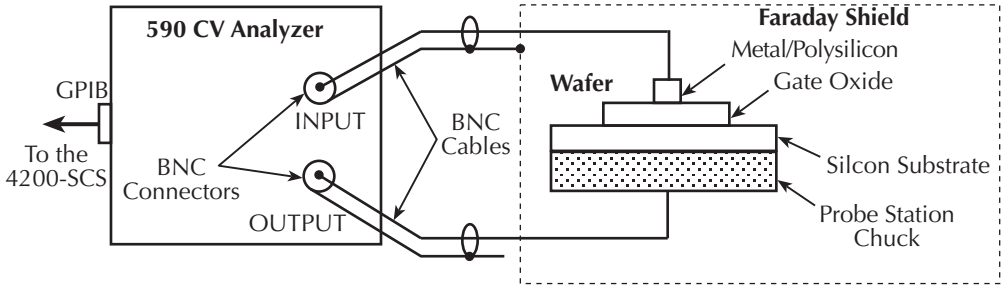
Connections

For details on Model 590 connections, see the Model 590 CV Analyzer Instruction Manual.

Signal connections

Basic signal connections for the Model 590 are shown in Figure C-2. The center conductors of the BNC connectors are connected to the device under test (DUT). The outer shield of one of the coaxial cables is typically connected to a Faraday shield. The Model 590 output is typically connected to the wafer backside (or well). The input is typically connected to the gate of a MOS-C.

Figure C-2
Basic Model 590 connections to the DUT



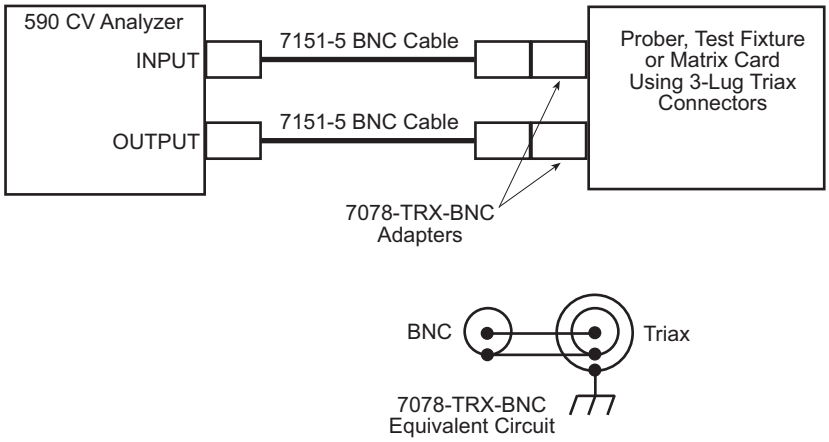
Triax connectors

Adapters are required to connect the Model 590 to equipment (for example, probe station, test fixture, matrix card) that use triax connectors. The Model 7087-TRX-BNC is a 3-lug triax to BNC adapter. As shown in Figure C-3, connect the adapters to the 3-slot triax connectors and then use a Model 7051-5 BNC cable to make the connections to the Model 590.

Figure C-3 also shows the equivalent circuit for the adapter.

NOTE: *See Using Switch Matrices in Appendix B for details on using a switch matrix with the Model 590 CV Analyzer.*

Figure C-3
Connecting the Model 590 to equipment using triax connectors



GPIB connections

The Model 4200-SCS controls the Model 590 via the General Purpose Interface Bus (GPIB). Use the Model 7007-1 or 7007-2 GPIB cable to connect the GPIB of the Model 590 to the GPIB of the Model 4200-SCS.

Cable compensation

Ideally, the Model 590 would only measure the capacitance of the DUT. However, signal pathways through the test cables, switch matrix, test fixture, and prober contribute unwanted capacitances that may adversely affect the measurement.

To correct for these unwanted capacitances, cable compensation should be performed before measuring the capacitance of DUT. In general, cable compensation is performed by connecting precisely known capacitance sources in place of the DUT and then measuring them. The Model 590 then uses these measured values to perform correction when measuring the DUT.

Cable compensation involves two steps:

1. The Model 590 calculates the compensation parameters based on the comparison between the given and measured values.
2. The Model 590 performs a probe-up offset measurement and suppresses any remaining offset capacitance. This step is performed every time a new measurement is performed.

Typically, cable compensation is performed for all four measurement ranges (2pF, 20 pF, 200 pF, and 2nF) of the Model 590. Once cable compensation is performed, it does not have to be done again unless the connection scheme to the DUT is changed or power is cycled.

For each measurement range of the Model 590, a low-capacitance source and a high-capacitance source must be used. Table C-1 lists the Model 5906 capacitance sources that can be used for each Model 590 range.

Table C-1
Model 5906 capacitance sources

590 range	Low capacitance source	High capacitance source
2pF	0.5 pF	1.5 pF
20 pF	4.7pF	18 pF
200 pF	47pF	180 pF
2 nF	470 pF	1.8nF

Cable compensation tests

The Model 4200-SCS has three user modules associated with cable compensation:

- **SaveCableCompCaps590**: Enter and save capacitance source values. The user enters the actual capacitance values of the capacitance sources. When the test is executed, the capacitance values are stored in a file at a user-specified directory path.
- **DisplayCableCompCaps590**: Places capacitance values into a spreadsheet. When this test is executed, the capacitance values saved by SaveCableCompCaps590 are placed into its **Sheet** tab.
- **CableCompensate590**: Performs cable compensation. The user specifies the ranges and test frequencies for cable compensation. When this test is executed, on-screen prompts will guide you through the cable compensation process.

CabCompFile: Each of the above three user modules for cable compensation use a cable compensation file to save / load capacitor source values. Therefore, all three user modules must use the same file directory path.

NOTE: *Details on all user modules for the Model 590 are provided in this appendix in ki590ulib user library reference.*

Using KCON to add Model 590 CV Analyzer to system

In order for the Model 4200-SCS to control an external instrument, that instrument must be added to the system configuration. The Model 590 CV Analyzer is added to the test system using the KCON (Keithley CONfiguration utility) as follows:

Step 1. Close KITE and open KCON

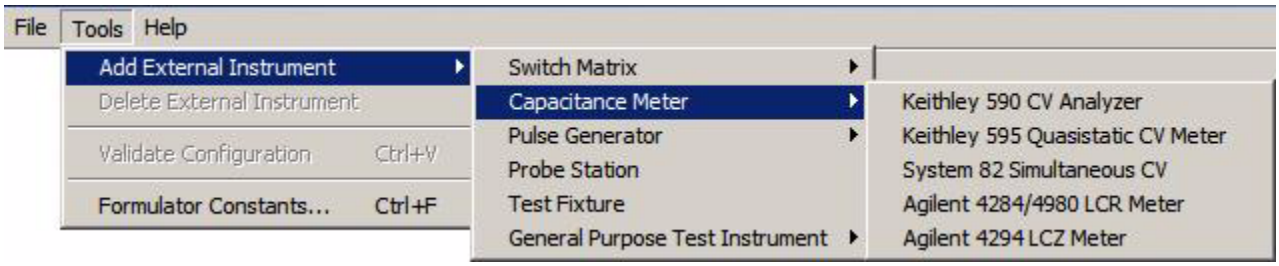
Close KITE by clicking the close button (X) at the top right-hand corner of the KITE panel. If you have made changes, you will be prompted to save them. On the windows desktop, double-click the **KCON** icon to open KCON.

For details on using KCON, see Keithley CONfiguration Utility (KCON) in Section 7.

Step 2. Add CV Analyzer

Add the Keithley Instruments Model 590 CV Analyzer from the **Tools** menu on the Toolbar of the KCON window (Figure C-4). Figure C-5 shows the **Properties and Connections** window for the CV Analyzer.

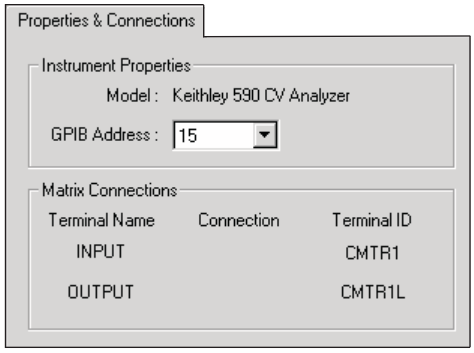
Figure C-4
KCON tools menu to add Model 590 CV Analyzer



Step 3. Set the GPIB address

The GPIB address setting in the **Instrument Properties** area of the Properties & Connections window (Figure C-5) must match the actual GPIB address of the Model 590. The address for the Model 590 is briefly displayed during its power-on sequence.

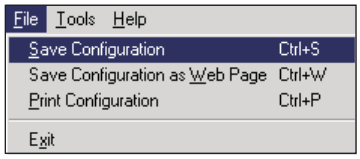
Figure C-5
Model 590 CV Analyzer Properties & Connections window



Step 4. Save the KCON configuration

The KCON configuration is saved from the **File** menu on the toolbar. As shown in Figure C-6, click **Save Configuration**.

Figure C-6
Save the KCON configuration



Step 5. Close KCON and open KITE

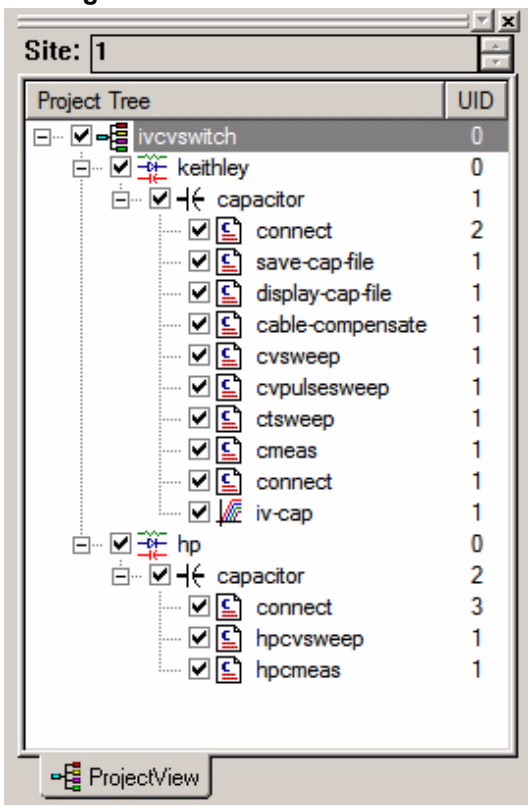
KCON can be closed from the **File** menu by clicking **Exit**. It can also be closed by clicking the **close** button (X) at the top right-hand corner of the KCON window.

On the windows desktop, double-click the **KITE** icon to open KITE.

Model 590 test examples

The test examples for the Model 590 CV Analyzer are controlled by user test modules (UTMs) in the **ivcvswitch** project. Figure C-7 shows the Project Navigator for the project. A switch matrix is not used for these examples.

Figure C-7
ivcvswitch Project Navigator



Example #1: Cable compensation

This example assumes that the Model 590 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a MOS-C on a wafer. Complete the following steps to perform cable compensation:

NOTE: *The three user modules for cable compensation must share the same file for capacitance source values. Therefore, the same file directory path must be used in all three user modules. For this example, use the default file directory path (see line 1 of the parameter list in Figure C-8, Figure C-9, and Figure C-11).*

Step A: Enter and save capacitance source values (SaveCableCompCaps590)

1. In the Project Navigator (Figure C-7), double-click **save-cap-file** to open the UTM. This UTM uses the SaveCableCompCaps590 user module. Figure C-8 shows the default parameter values. Keep in mind that these values are only listed to show typical capacitance values.

Figure C-8
SaveCableCompCaps590 default parameters

Formulator
User Libraries: KI590ulib
User Modules: SaveCableCompCaps590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	Lo2p100k	Input	DOUBLE	0.4e-12
3	Lo2p1M	Input	DOUBLE	0.4e-12
4	Hi2p100k	Input	DOUBLE	1.8e-12
5	Hi2p1M	Input	DOUBLE	1.8e-12
6	Lo20p100k	Input	DOUBLE	4e-12
7	Lo20p1M	Input	DOUBLE	4e-12
8	Hi20p100k	Input	DOUBLE	18e-12
9	Hi20p1M	Input	DOUBLE	18e-12
10	Lo200p100k	Input	DOUBLE	40e-12
11	Lo200p1M	Input	DOUBLE	40e-12
12	Hi200p100k	Input	DOUBLE	180e-12
13	Hi200p1M	Input	DOUBLE	180e-12
14	Lo2n100k	Input	DOUBLE	400e-12
15	Lo2n1M	Input	DOUBLE	400e-12
16	Hi2n100k	Input	DOUBLE	1800e-12
17	Hi2n1M	Input	DOUBLE	1800e-12

2. In the parameter list, enter the capacitance source calibration value for each range and frequency. If using the Model 5906, each capacitor has a label indicating the calibration value at 100 kHz and at 1 MHz.

   For example, assume the low capacitance source for the 2 pF range is 0.47773 pF (100 kHz) and 0.47786 pF (1 MHz). Enter these values using scientific notation:

   Line 2 (Lo2p100k) - Enter 0.47773e-12

   Line 3 (Lo2p1M) - Enter 0.47786e-12

3. In the Project Navigator, click **save-cap-file** to select the UTM, and then click the green run key to execute the test. The capacitor source values entered into the UTM will be saved in the file using the directory path specified in line 1 of the parameter list.

Step B: Place capacitance source values in a spreadsheet (DisplayCableCompCaps590)

1. In the Project Navigator (Figure C-7), double-click **display-cap-file** to open the UTM. Figure C-9 shows the parameter list for the DisplayCableCompCaps590 user module.
2. Ensure that line 1 of the parameter list has the same file directory path that is used in Step 1 (Figure C-8). Lines 3, 5, and 8 set array size. These must be set to 8 as shown in Figure C-9.

Figure C-9
DisplayCableCompCaps590 default parameters

Formulator
User Libraries: KI590ulib
User Modules: DisplayCableCompCaps590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	Range	Output	DBL_ARRAY	
3	RangeSize	Input	INT	8
4	Values100k	Output	DBL_ARRAY	
5	Values100kSize	Input	INT	8
6	Values1M	Output	DBL_ARRAY	
7	Values1MSize	Input	INT	8

3. Execute the **display-cap-file** UTM by clicking the green **Run** button. The calibration source values entered and saved in Step A are placed into its spreadsheet.
4. In the workspace, click the **Sheet** tab for **display-cap-file** to display its spreadsheet. An example spreadsheet is shown in Figure C-10.

Figure C-10
Display-cap-file spreadsheet showing capacitor source values

	A	B	C	D
1	DisplayCable	Range	Values100k	Values1M
2	0	2.00000E-12	4.77700E-13	4.77700E-13
3		2.00000E-12	1.46100E-12	1.46100E-12
4		2.00000E-11	4.79600E-12	4.79600E-12
5		2.00000E-11	1.78300E-11	1.78300E-11
6		2.00000E-10	4.66800E-11	4.66800E-11
7		2.00000E-10	1.81100E-10	1.81100E-10
8		2.00000E-09	4.75500E-10	4.75900E-10
9		2.00000E-09	1.77100E-09	1.77600E-09
10				

Step C: Perform cable compensation (CableCompensate)

1. In the Project Navigator (Figure C-7), double-click **cable-compensate** to open the UTM. Figure C-11 shows the default parameters for the CableCompensate590 user module.
2. Make sure line 1 of the parameter list has the same file directory path that is used in Step 1 (Figure C-8).
3. Enable / disable cable compensation. Lines 5 through 10 of the parameter list are used to either disable (0) or enable (1) cable compensation for the test frequencies and ranges. Figure C-11 shows cable compensation enabled for all ranges and test frequencies.

Figure C-11
CableCompensate590 default parameters

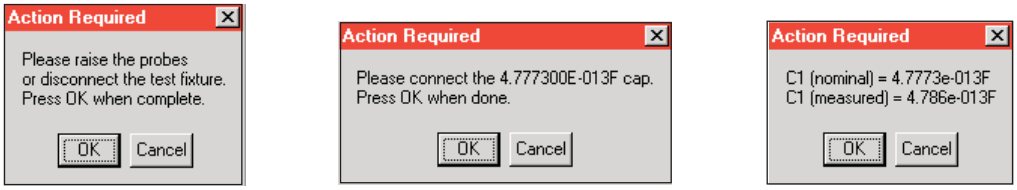
Formulator
User Libraries: KI590ulib
User Modules: CableCompensate590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	Freq100k	Input	INT	1
6	Freq1M	Input	INT	1
7	Range2p	Input	INT	1
8	Range20p	Input	INT	1
9	Range200p	Input	INT	1
10	Range2n	Input	INT	1

4. Execute the **cable-compensate** UTM by clicking the green **Run** button. A series of dialog boxes will guide you through the cable compensation process. The three basic dialog boxes are shown in Figure C-12:
   - **Figure C-12A**: This dialog box will appear when an offset (open circuit) measurement is required. Open the circuit as close to the DUT as possible.
   - **Figure C-12B**: This dialog box will instruct you to connect a capacitance source in place of the DUT. Note that the value in the dialog box corresponds to a calibration value entered by the user in Step 1. Connect the capacitance source as close to the DUT as possible.
   - **Figure C-12C**: This dialog box compares the measured value to the calibration (nominal) value entered by the user. The two readings should be fairly close. If they are not, you probably connected the wrong capacitance source or had an open circuit condition. In that case, click **Cancel** to abort the cable compensation process.

NOTE: *Clicking **Cancel** in a cable compensation dialog box aborts the cable compensation process. You can start over by clicking the green **Run** button.*

Figure C-12
Cable compensation dialog boxes



A. Measure offset B. Measure capacitance source C. Compare readings

Example #2: C-V sweep

This example assumes that the Model 590 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a MOS-C on a wafer. Complete the following steps to perform a C-V sweep:

Open and execute cvsweep UTM

1. In the Project Navigator (Figure C-7), double-click **cvsweep** to open the UTM (Figure C-13). You can modify the test parameters from the **Definition** tab, if desired. If you use the default parameters shown in Figure C-13, the Model 590 will perform a -4V to +6V staircase sweep using 50 mV steps.
2. Execute the test by clicking the green **Run** button.

cvsweep test description

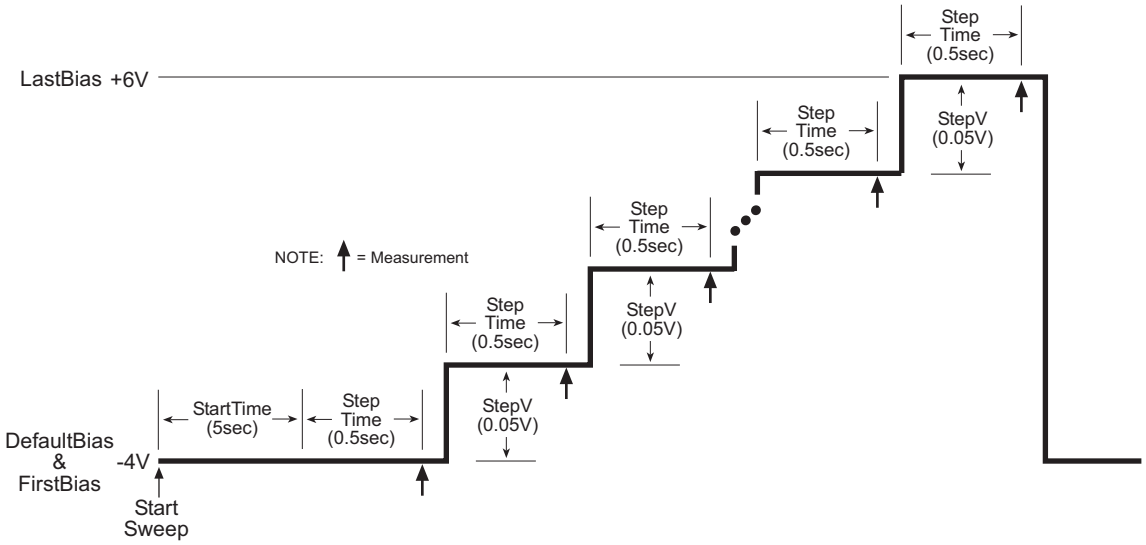
As shown in Figure C-13, the **cvsweep** UTM uses the CvSweep590 user module. For details on this test description, see the reference information for the CvSweep590 user module, ki590ulib user library reference, in this appendix. Figure C-13 shows the complete default parameter list for the test, while Figure C-14 shows the configured sweep.

Figure C-13
CvSweep590 user module (cvsweep UTM)

Formulator | User Libraries: KI590ulib
Output Values | User Modules: CvSweep590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	OffsetCorrect	Input	INT	0
6	Waveform	Input	INT	1
7	FirstBias	Input	DOUBLE	-4.0000E+0
8	LastBias	Input	DOUBLE	6.0000E+0
9	StepV	Input	DOUBLE	0.05
10	Frequency	Input	INT	0
11	DefaultBias	Input	DOUBLE	-4.0000E+0
12	StartTime	Input	DOUBLE	5.0000E+0
13	StepTime	Input	DOUBLE	500.0000E-3
14	Range	Input	DOUBLE	2.0000E-9
15	Model	Input	INT	0
16	Filter	Input	INT	1
17	ReadingRate	Input	INT	3
18	C	Output	DBL_ARRAY	
19	Csize	Input	INT	1350
20	V	Output	DBL_ARRAY	
21	Vsize	Input	INT	1350
22	G_or_R	Output	DBL_ARRAY	
23	G_or_Rsize	Input	INT	1350
24	T	Output	DBL_ARRAY	
25	Tsize	Input	INT	1350

Figure C-14
C-V linear staircase sweep



ki590ulib user library reference

The user modules in the ki590ulib user library are used to control the Model 590 CV Analyzer. These user modules are summarized in Table C-2. Also listed in the table are the Keithley Instruments-created UTM names that utilize the user modules.

NOTE: *Details for each of the user modules follow Table C-2. Specific user-entered parameters for each user module are explained under the heading, User module description, with detailed information or possible user inputs listed under the "INPUTS" heading in each user-model description.*

Table C-2
ki590ulib user library

User module	UTM name	Description
CableCompensate590	cable-compensate	Performs cable compensation using known capacitance source values.
Cmeas590	cmeas	Performs a single capacitance measurement.
CtSweep590	ctsweep	Performs a capacitance vs. time measurements.
CvPulseSweep590	cvpulsesweep	Performs capacitance vs. voltage measurements using a pulse sweep.
CvSweep590	cvsweep	Performs capacitance vs. voltage measurements using a staircase sweep.
DisplayCableCompCaps590	display-cap-file	Places capacitance source values in a spreadsheet.
SaveCableCompCaps590	save-cap-file	Saves entered capacitance source values in a file.

CableCompensate590 user module

Overview

This user module is used to perform cable compensation for the selected ranges and test frequencies of the Model 590. Figure C-15 shows the default input parameters for a UTM that uses the CableCompensate590 user module.

For the input parameters shown in Figure C-15, cable compensation for the Model 590 will be performed for the 2 pF, 20 pF, 200 pF, and 2 nF ranges. It will be performed for both the 100 kHz and 1 MHz test frequencies. The line 1 input parameter indicates the directory path where the user-input capacitor source values are saved. These values are entered and saved using the SaveCableCompCaps590 user module.

Test example #1 demonstrates how cable compensation is performed (see Model 590 test examples earlier in this appendix).

Figure C-15
CableCompensate590 (default parameters)

Formulator

User Libraries: KI590ulib

User Modules: CableCompensate590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	Freq100k	Input	INT	1
6	Freq1M	Input	INT	1
7	Range2p	Input	INT	1
8	Range20p	Input	INT	1
9	Range200p	Input	INT	1
10	Range2n	Input	INT	1

User module description

The CableCompensate590 routine performs the ki590 cable compensation procedure using the capacitor values stored in the specified cable compensation file. The resultant compensation values generated by the compensation process are stored in the same file.

Syntax:

status = CableCompensate590(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int Freq100 k, int Freq1 M, int Range2 p, int Range20 p, int Range200 p, int range2 n);

PROCEDURE:

For each range and test frequency specified by the input parameters, the following will occur:

1. You will be prompted to open the circuit so that an offset capacitance measurement can be made.
2. Once the offset capacitance measurement is completed, you will be prompted to connect the low value capacitor for the selected range. The system will perform the low value capacitor compensation.
3. Next, you will be prompted to connect the high value capacitor for the selected range. The system will perform the high value capacitor compensation.
4. You will then be prompted to reconnect the low capacitor.

5. The nominal and measured values will be displayed in a dialog box. If you are unsatisfied with the measurement, press **cancel** to abort the procedure. If you press **cancel**, the cable compensation file will not be affected.

When all selected ranges and frequencies have been compensated successfully, the cable compensation values will be saved.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist, or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the ki590's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following ***NOTE***.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to "0."*

OutPin (int) The DUT pin to which the ki590's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous ***NOTE***.

Freq100k (int) A flag indicating whether to perform the compensation procedure for the 100 kHz frequency. A value of 0 will skip the compensation procedure for this frequency, while a value of 1 will perform the compensation procedure for this frequency.

Freq1M (int) A flag indicating whether to perform the compensation procedure for the 1 MHz frequency. A value of 0 will skip the compensation procedure for this frequency, while a value of 1 will perform the compensation procedure for this frequency.

Range2p (int) A flag indicating whether to perform the compensation procedure for the 2pF range. A value of 1 will perform compensation for the 2pF range, while a value of 0 will skip compensation for the 2 pF range.

Range20p (int) A flag indicating whether to perform the compensation procedure for the 20 pF range. A value of 1 will perform compensation for the 20 pF range, while a value of 0 will skip compensation for the 20 pF range.

Range200p (int) A flag indicating whether to perform the compensation procedure for the 200 pF range. A value of 1 will perform compensation for the 200 pF range, while a value of 0 will skip compensation for the 200 pF range.

Range2n (int) A flag indicating whether to perform the compensation procedure for the 2 nF range. A value of 1 will perform compensation for the 2 nF range, while a value of 0 will skip compensation for the 2 nF range.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (located on the **Sheet** tab).

0 OK

-**10000** (INVAL_INST_ID) The specified instrument ID does not exist.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

Cmeas590 user module

Overview

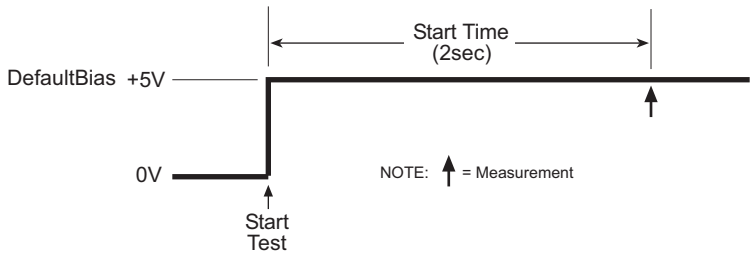
This user module is used to perform a single, fixed-bias capacitance and conductance measurement. Figure C-16 shows the default parameters for the **cmeas** UTM, which uses the Cmeas590 user module. In general, the Model 590 is set to source 5 V for 2 seconds, then perform a measurement, as shown in Figure C-17.

Figure C-16
Cmeas590 (cmeas UTM parameters)

Formulator | User Libraries: KI590ulib
Output Values | User Modules: Cmeas590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	OffsetCorrect	Input	INT	0
6	Frequency	Input	INT	0
7	DefaultBias	Input	DOUBLE	5.0000E+0
8	StartTime	Input	DOUBLE	2.0000E+0
9	Range	Input	DOUBLE	200e-12
10	Model	Input	INT	0
11	Filter	Input	INT	0
12	ReadingRate	Input	INT	3
13	C	Output	DOUBLE_P	
14	V	Output	DOUBLE_P	
15	G_or_R	Output	DOUBLE_P	

Figure C-17
Cmeas590 measurement



User module description

The Cmeas590 routine measures capacitance and conductance using the Keithley Instruments Model 590 CV Analyzer. If desired, an offset correction measurement is taken and cable compensation can be used.

Syntax:

status = Cmeas590(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int OffsetCorrect, int Frequency, double DefaultBias, double StartTime, double Range, int Model, int Filter, int ReadingRate, double *C, double *V, double *G_or_R);

PROCEDURE:

1. You will be prompted to open the circuit so that an offset capacitance measurement can be made, if desired.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.
3. The capacitance and conductance will then be measured.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the ki590's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following ***NOTE***.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to "0."*

OutPin (int) The DUT pin to which the ki590's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous ***NOTE***.

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, then no offset measurement will be taken. If OffsetCorrect is 1, then an offset measurement will be made.

Frequency (int) A variable that selects the measurement frequency to use. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

DefaultBias (double) The DC bias to use for the measurement. Valid inputs are -20 V to +20 V.

StartTime (double) The amount of time to delay after applying the DefaultBias voltage step. The valid range of inputs is 0.001 seconds to 65 seconds.

Range (double) The measurement range to use. The valid ranges of range values are 2 e-12, 20 e-12, 200 e-12, and 2 e-9 F:

Table C-3
Cmeas590 valid measurement range values

Range	100 kHz	1 MHz
1	2 pF / 2 uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2 mS
4	2 nF / 2 mS	2 nF / 20 mS

Model (int) Which measurement model to use. Entering 0 for this parameter will select the parallel model, while value of 1 will select the series model.

Filter (int) Enable / disable the analog filter. The analog filter can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time. Entering a value of 0 for this parameter will disable the filter; a value of 1 will enable the filter.

ReadingRate (int) Select the reading rate used to acquire the measurements. Valid input values are 0 through 4, as shown below:

Table C-4
Cmeas590 ReadingRate valid input values

Reading rate	Nominal reading rate	Display readings	Resolution
0	1000 / sec	C only	3.5 digits
1	75 / sec	C,G,V	3.5 digits
2	18 / sec	C,G,V	4.5 digits
3	10 / sec	C,G,V	4.5 digits
4	1 / sec	C,G,V	4.5 digits

OUTPUTS:

C (double *) The measured capacitance.

V (double *) The bias voltage used.

G_or_R (double *) If the parallel Model (0) is selected, G_or_R is the measured conductance. If the series Model (1) is selected, G_or_R is the measured resistance.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0 OK.

-**10000** (INVAL_INST_ID) The specified instrument ID does not exist.

-**10020** (COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

-**10023** (KI590_MEAS_ERROR) A measurement error occurred.

-**10090** (GPIB_ERROR_OCCURED) A GPIB communications error occurred.

-10091 (GPIB_TIMEOUT) A time-out occurred during communications.

-10102 (ERROR_PARSING) There was an error parsing the Model 590's response.

-10104 (USER_CANCEL) The user CANCELED the correction procedure.

CtSweep590 user module

Overview

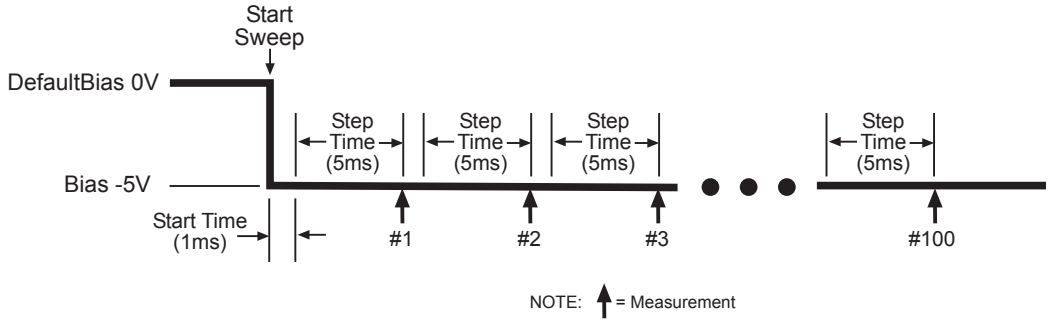
This user module performs a capacitance versus time sweep. Figure C-18 shows the default parameters for the **ctsweep** UTM, which uses the CtSweep590 user module. In this example, the Model 590 is set to source -5 V and performs 100 capacitance measurements using a 5 ms time interval, as shown in Figure C-19.

Figure C-18
CtSweep590 (ctsweep UTM parameters)

Formulator | User Libraries: KI590ulib
Output Values | User Modules: CtSweep590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	OffsetCorrect	Input	INT	0
6	Frequency	Input	INT	1
7	DefaultBias	Input	DOUBLE	-5.0000E+0
8	Bias	Input	DOUBLE	0
9	StartTime	Input	DOUBLE	1.0000E-3
10	StepTime	Input	DOUBLE	5.0000E-3
11	Range	Input	DOUBLE	2e-9
12	Model	Input	INT	0
13	Filter	Input	INT	1
14	Count	Input	INT	100
15	ReadingRate	Input	INT	3
16	C	Output	DBL_ARRAY	
17	Csize	Input	INT	1350
18	G_or_R	Output	DBL_ARRAY	
19	G_or_Rsize	Input	INT	1350
20	T	Output	DBL_ARRAY	
21	Tsize	Input	INT	1350

Figure C-19
C-t measurements



User module description

The CtSweep590 routine performs a capacitance vs. time (Ct) sweep using the Keithley Instruments Model 590 CV Analyzer. If desired, an offset correction measurement is taken and the cable compensation can be used.

Syntax:

status = int CtSweep590(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int OffsetCorrect, int Frequency, double DefaultBias, double Bias, double StartTime, double StepTime, double Range, int Model, int Filter, int Count, int ReadingRate, double *C, int Csize, double *G_or_R, int G_or_Rsize, double *T, int Tsize)

PROCEDURE:

1. You will be prompted to open the circuit so that an offset capacitance measurement can be made, if desired.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.
3. A C-t sweep is performed.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the ki590's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following ***NOTE***.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to "0."*

OutPin (int) The DUT pin to which the ki590's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous ***NOTE***.

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, no offset measurement will be taken. If OffsetCorrect is 1, an offset measurement will be made.

Frequency (int) A variable that selects the measurement frequency to use. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

DefaultBias (double) The DC bias applied before and after a sweep. Valid ranges of input are -20 V to +20 V.

Bias (double) The DC bias applied during a sweep. Valid inputs are -20 V to +20 V.

StartTime (double) The time period occurring on the first bias step, from the point the instrument is first triggered until the first step time. The valid range of inputs is 0.001 seconds to 65 seconds.

StepTime (double) The time period after a transition to a new bias step and before the instrument begins a measurement. The valid range of inputs is 0.001 seconds to 65 seconds.

Range (double) The measurement range to use. The valid ranges of range values are 2 e-12, 20 e-12, 200 e-12, and 2 e-9 F:

Table C-5
CtSweep590 valid measurement range values

Range	100 kHz	1 MHz
1	2pF / 2uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2 mS
4	2 nF / 2 mS	2 nF / 20 mS

Model (int) Which measurement model to use. Entering 0 for this parameter will select the parallel model, while 1 will select the series model.

Filter (int) Enable / disable the analog filter. The analog filter can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time. Entering 0 for this parameter will disable the filter; 1 will enable the filter.

Count (int) The number of readings per sweep. The valid range of input is from 1 through 1350.

ReadingRate (int) Selects the reading rate used to acquire the measurements. Valid ranges of input are 0 through 4 as shown below:

Table C-6
CtSweep590 valid ReadingRate range values

Reading rate	Nominal reading rate	Readings	Display resolution
0	1000 / sec	C only	3.5 digits
1	75 / sec	C,G,V	3.5 digits
2	18 / sec	C,G,V	4.5 digits
3	10 / sec	C,G,V	4.5 digits
4	1 / sec	C,G,V	4.5 digits

Csize, **G_or_Rsize**, **Tsize** (int) These values **must** be set equal to a value that at minimum is equal to the Count. If in doubt, set this value to 1350.
The values of these parameters are set to 1350 when they are called from KITE UTM's.

OUTPUTS:

C (double *) The measured array of capacitance values.

G_or_R (double *) If the parallel model (0) is selected, G_or_R is the measured array of conductance values. For the series model (1), G_or_R is the measured array of resistance values.

T (double *) The array of time stamps for each measurement step.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0	OK.
-**10000**	(INVAL_INST_ID) The specified instrument ID does not exist.
-**10020**	(COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.
-**10021**	(COMP_FILE_NOT_EXIST) The specified compensation file does not exist.
-**10022**	(KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.
-**10023**	(KI590_MEAS_ERROR) A measurement error occurred.
-**10090**	(GPIB_ERROR_OCCURRED) A GPIB communications error occurred.
-**10091**	(GPIB_TIMEOUT) A time-out occurred during communications.
-**10101**	(ARRAY_SIZE_TOO_SMALL) The specified value for Csize, G_or_Rsize, Vsize, or Tsize was too small for the number of steps in the sweep.
-**10102**	(ERROR_PARSING) There was an error parsing the Model 590's response.
-**10104**	(USER_CANCEL) The user CANCELED the correction procedure.

CvPulseSweep590 user module

Overview

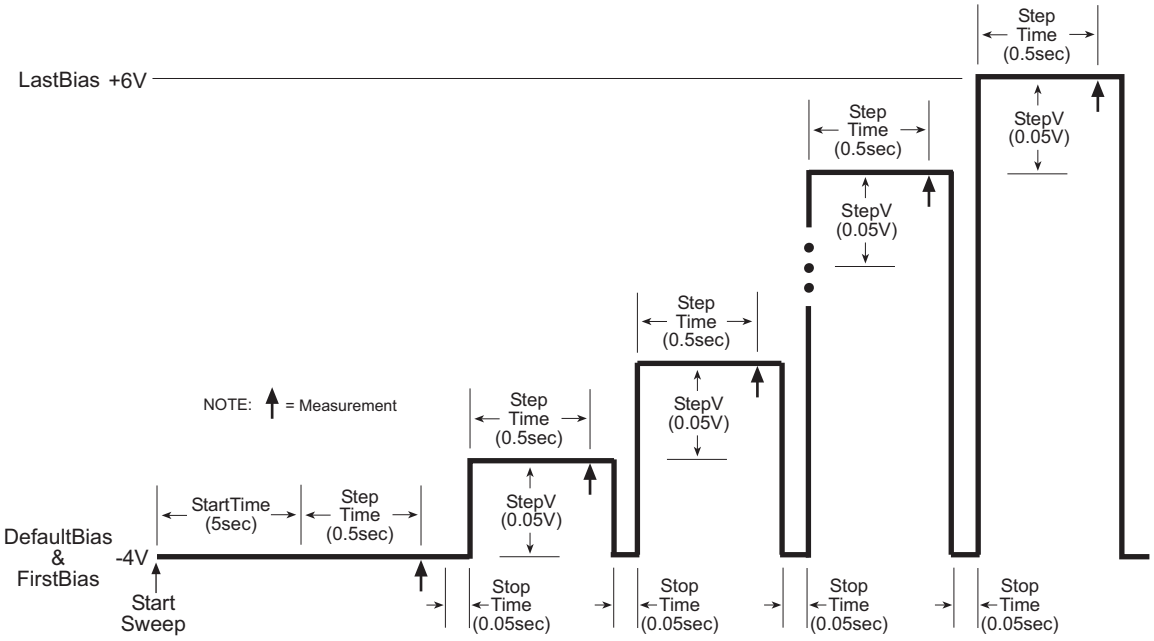
This user module performs a capacitance versus voltage pulse sweep. Figure C-20 shows the default parameters for the **cvpulsesweep** UTM, which uses the CvPulseSweep590 user module. In this example, the Model 590 outputs a series of pulses in 50 mV steps from -4V to +6V. As shown in Figure C-21, a measurement is performed on each pulse step.

Figure C-20
CvPulseSweep590 (cvpulsesweep UTM parameters)

Formulator | User Libraries: KI590ulib
Output Values | User Modules: CvPulseSweep590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	OffsetCorrect	Input	INT	0
6	FirstBias	Input	DOUBLE	-4.0000E+0
7	LastBias	Input	DOUBLE	6.0000E+0
8	StepV	Input	DOUBLE	0.05
9	Frequency	Input	INT	0
10	DefaultBias	Input	DOUBLE	-4.0000E+0
11	StartTime	Input	DOUBLE	5.0000E+0
12	StopTime	Input	DOUBLE	0.05
13	StepTime	Input	DOUBLE	500.0000E-3
14	Range	Input	DOUBLE	2.0000E-9
15	Model	Input	INT	0
16	Filter	Input	INT	0
17	ReadingRate	Input	INT	3
18	C	Output	DBL_ARRAY	
19	Csize	Input	INT	1350
20	V	Output	DBL_ARRAY	
21	Vsize	Input	INT	1350
22	G_or_R	Output	DBL_ARRAY	
23	G_or_Rsize	Input	INT	1350
24	T	Output	DBL_ARRAY	
25	Tsize	Input	INT	1350

Figure C-21
C-V pulse-sweep measurements



User module description

The CvPulseSweep590 routine performs a capacitance vs. voltage (C-V) sweep using the pulse waveform capability of the Keithley Instruments Model 590 CV Analyzer. If desired, an offset correction measurement is taken and the cable compensation can be used.

Syntax:

status = CvPulseSweep590(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int OffsetCorrect, double FirstBias, double LastBias, double StepV, int Frequency, double DefaultBias, double StartTime, double StepTime, double Range, int Model, int Filter, int ReadingRate, double *C, int Csize, double *V, int Vsize, double *G_or_R, int G_or_Rsize, double *T, int Tsize);

PROCEDURE:

1. You will be prompted to open the circuit so that an offset capacitance can be made, if desired.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.
3. A CV pulse sweep is taken.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the ki590's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following ***NOTE***.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to "0."*

OutPin (int) The DUT pin to which the ki590's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous ***NOTE***.

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, no offset measurement will be taken. If OffsetCorrect is 1, an offset measurement will be made.

FirstBias (double) The starting voltage for the sweep. Valid values for this parameter range from -20 V to +20 V.

LastBias (double) The last voltage used in the sweep. Valid values for this parameter range from -20 V to +20 V.

StepV (double) The voltage step size. The valid range of inputs for this parameter is -20 V to +20 V. The value of ((LastBias - FirstBias) / StepV) +1 must be less than or equal to the Csize, Vsize, G_or_Rsize, and Tsize parameters.

Frequency (int) A variable that selects the measurement frequency to use. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

DefaultBias (double) The DC bias applied before and after a sweep. Valid inputs are -20 V to +20 V.

StartTime (double) The time period occurring on the first bias step, from the point the instrument is first triggered until the first step time. The valid range of inputs is 0.001 seconds to 65 seconds.

StopTime (double) The time period between pulses with the Model 590 at the default bias. The valid range of inputs is 0.001 seconds to 65 seconds.

StepTime (double) The time period after a transition to a new bias step and before the instrument begins a measurement. The valid range of inputs is 0.001 seconds to 65 seconds.

Range (double) The measurement range to use. The valid ranges of range values are 2 e-12, 20 e-12, 200 e-12, and 2 e-9 F:

Table C-7
CvPulseSweep590 valid measurement range values

Range	100 kHz	1 MHz
1	2 pF / 2 uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2 mS
4	2 nF / 2 mS	2 nF / 20 mS

Model (int) Which measurement model to use. Entering 0 for this parameter will select the parallel model, while 1 will select the series model.

Filter (int) Enable / disable the analog filter. The analog filter can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time. Entering 0 for this parameter will disable the filter; 1 will enable the filter.

ReadingRate (int) Selects the reading rate used to acquire the measurements. Valid inputs are 0 through 4 as shown below:

Table C-8
CvPulseSweep590 valid ReadingRate range values

Reading rate	Nominal reading rate	Readings	Display resolution
0	1000 / sec	C only	3.5 digits
1	75 / sec	C,G,V	3.5 digits
2	18 / sec	C,G,V	4.5 digits
3	10 / sec	C,G,V	4.5 digits
4	1 / sec	C,G,V	4.5 digits

Csize, Vsize, G_or_Rsize, Tsize (int) These parameters **must** be set to a value that is equal to or greater than the number of voltage steps in the sweep, or = ((LastBias - FirstBias) / StepV) + 1. When this function is called from a KITE UTM, these parameters are all fixed to 1350.

OUTPUTS:

C (double *) The measured array of capacitance values.

V (double *) The array of bias voltages used.

G_or_R (double *) If the parallel model (0) is selected, G_or_R is the measured array of conductance values. For the series model (1), G_or_R is the measured array of resistance values.

T (double *) The array of time stamps for each measurement step.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0 OK.

-**10000** (INVAL_INST_ID) The specified instrument ID does not exist.

-**10020** (COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

-**10023** (KI590_MEAS_ERROR) A measurement error occurred.

-**10090** (GPIB_ERROR_OCCURED) A GPIB communications error occurred.

-**10091** (GPIB_TIMEOUT) A time-out occurred during communications.

-**10101** (ARRAY_SIZE_TOO_SMALL) The specified value for Csize, G_or_Rsize, Vsize, or Tsize was too small for the number of steps in the sweep.

-**10102** (ERROR_PARSING) There was an error parsing the Model 590's response.

-**10104** (USER_CANCEL) The user CANCELED the correction procedure.

CvSweep590 user module

Overview

This user module performs a capacitance versus voltage staircase sweep. Figure C-22 shows the default parameters for the **cvsweep** UTM, which uses the CvSweep590 user module. In general, the Model 590 outputs a linear staircase voltage sweep from -4V to +6V in 50 mV steps. As shown in Figure C-23, a capacitance measurement is performed on each step of the sweep.

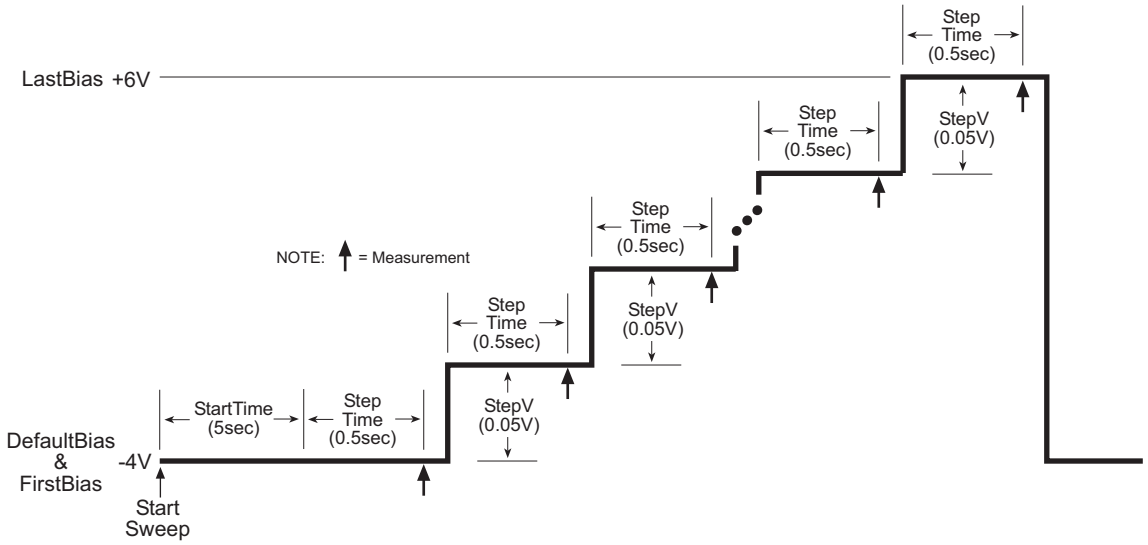
A test example demonstrates how to perform a C-V sweep (see Model 590 test examples earlier in this appendix).

Figure C-22
CvSweep590 (cvsweep UTM parameters)

Formulator | User Libraries: KI590ulib
Output Values | User Modules: CvSweep590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	OffsetCorrect	Input	INT	0
6	Waveform	Input	INT	1
7	FirstBias	Input	DOUBLE	-4.0000E+0
8	LastBias	Input	DOUBLE	6.0000E+0
9	StepV	Input	DOUBLE	0.05
10	Frequency	Input	INT	0
11	DefaultBias	Input	DOUBLE	-4.0000E+0
12	StartTime	Input	DOUBLE	5.0000E+0
13	StepTime	Input	DOUBLE	500.0000E-3
14	Range	Input	DOUBLE	2.0000E-9
15	Model	Input	INT	0
16	Filter	Input	INT	1
17	ReadingRate	Input	INT	3
18	C	Output	DBL_ARRAY	
19	Csize	Input	INT	1350
20	V	Output	DBL_ARRAY	
21	Vsize	Input	INT	1350
22	G_or_R	Output	DBL_ARRAY	
23	G_or_Rsize	Input	INT	1350
24	T	Output	DBL_ARRAY	
25	Tsize	Input	INT	1350

Figure C-23
C-V sweep measurements



User module description

The CvSweep590 routine performs a capacitance vs. voltage (C-V) sweep using the Keithley Instruments Model 590 CV Analyzer. If desired, an offset correction measurement is taken and the cable compensation can be used.

Syntax:

status = CvSweep590(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int OffsetCorrect, int Waveform, double FirstBias, double LastBias, double StepV, int Frequency, double DefaultBias, double StartTime, double StepTime, double Range, int Model, int Filter, int ReadingRate, double *C, int Csize, double *V, int Vsize, double *G_or_R, int G_or_Rsize, double *T, int Tsize);

PROCEDURE:

1. You will be prompted to open the circuit so that an offset capacitance measurement can be made, if desired.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.
3. A CV sweep is taken.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the ki590's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the

Model 590 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following ***NOTE***.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to "0."*

OutPin (int) The DUT pin to which the ki590's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 590 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous ***NOTE***.

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, no offset measurement will be taken. If OffsetCorrect is 1, an offset measurement will be made.

Waveform (int) Selects either the single staircase or dual staircase waveform; 1 selects single, 2 selects dual.

FirstBias (double) The starting voltage for the sweep. Valid values for this parameter range from -20 V to +20 V.

LastBias (double) The last voltage used in the sweep. Valid values for this parameter range from -20 V to +20 V.

StepV (double) The voltage step size. The valid range of inputs for this parameter is -20 V to +20 V. The value of ((LastBias - FirstBias) / StepV) + 1 must be less than or equal to the Csize, Vsize, G_or_Rsize, and Tsize parameters.

Frequency (int) A variable that selects the measurement frequency to use. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

DefaultBias (double) The DC bias applied before and after a sweep. Valid inputs are -20 V to +20 V.

StartTime (double) The time period occurring on the first bias step, from the point the instrument is first triggered until the first step time. The valid range of inputs is 0.001 seconds to 65 seconds.

StepTime (double) The time period after a transition to a new bias step and before the instrument begins a measurement. The valid range of inputs is 0.001 seconds to 65 seconds.

Range (double) The measurement range to use. The valid ranges of range values are 2 e-12, 20 e-12, 200 e-12, and 2 e-9 F:

Table C-9
CvSweep590 valid measurement range values

Range	100 kHz	1 MHz
1	2 pF / 2 uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2 mS
4	2 nF / 2 mS	2 nF / 20 mS

Model (int) Which measurement model to use. Entering 0 for this parameter will select the parallel model, while 1 will select the series model.

Filter (int) Enable / disable the analog filter. The analog filter can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time. Entering 0 for this parameter will disable the filter; 1 will enable the filter.

ReadingRate (int) Selects the reading rate used to acquire the measurements. Valid inputs are 0 through 4 as shown below:

Table C-10
CvSweep590 valid ReadingRate range values

Reading rate	Nominal reading rate	Readings	Display resolution
0	1000 / sec	C only	3.5 digits
1	75 / sec	C,G,V	3.5 digits
2	18 / sec	C,G,V	4.5 digits
3	10 / sec	C,G,V	4.5 digits
4	1 / sec	C,G,V	4.5 digits

Csize, Vsize (int) These parameters **must** be set to a value equal to or greater than the number of voltage steps in the sweep, or = ((LastBias - FirstBias) / StepV) + 1.

G_or_Rsize, Tsize When this function is called from a KITE UTM, these parameters are all fixed to 1350.

OUTPUTS:

C **(double *) The measured array of capacitance values.**

V (double *) The array of bias voltages used.

G_or_R (double *) If the parallel Model (0) is selected, G_or_R is the measured array of conductance values. For the series Model (1), G_or_R is the measured array of resistance values.

T (double *) The array of time stamps for each measurement step.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0 OK.

-**10000** (INVAL_INST_ID) The specified instrument ID does not exist.

-**10020** (COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

-**10023** (KI590_MEAS_ERROR) A measurement error occurred.

-**10090** (GPIB_ERROR_OCCURED) A GPIB communications error occurred.

-**10091** (GPIB_TIMEOUT) A time-out occurred during communications.

-**10101** (ARRAY_SIZE_TOO_SMALL) The specified value for Csize, G_or_Rsize, Vsize, or Tsize was too small for the number of steps in the sweep.

-**10102** (ERROR_PARSING) There was an error parsing the Model 590's response.

-**10104** (USER_CANCEL) The user CANCELED the correction procedure.

DisplayCableCompCaps590 user module

Overview

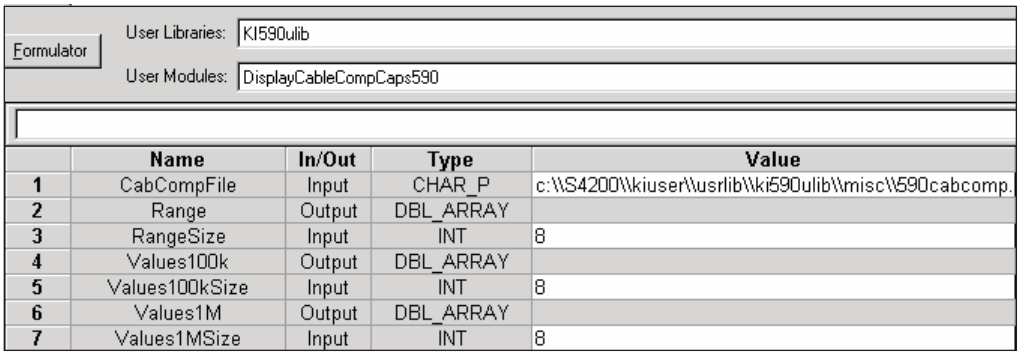
This user module is used for Model 590 cable compensation. When this test is run, the nominal capacitance source values saved by the SaveCableCompCaps590 user module, are placed into a spread sheet for convenient viewing.

The default parameters for this user module are shown in Figure C-24. Line 1 specifies the file directory path where the capacitance values are saved. This file directory path must be the same as the one used by the SameCableCompCaps590 user module.

To prevent unpredictable results, the array size values for the Range, 100 k and 1 M arrays (lines 3, 5, and 7) must be set to 8 as shown in Figure C-24.

Test example #1 demonstrates how cable compensation is performed (see Model 590 test examples earlier in this section).

Figure C-24
DisplayCableCompCap590 (default parameters)



	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	Range	Output	DBL_ARRAY	
3	RangeSize	Input	INT	8
4	Values100k	Output	DBL_ARRAY	
5	Values100kSize	Input	INT	8
6	Values1M	Output	DBL_ARRAY	
7	Values1MSize	Input	INT	8

User module description

The DisplayCableCompCaps590 reads the nominal cable compensation values that are stored in the compensation file, and returns them to the calling function (or in the case of KITE, to the UTM data sheet). The returned arrays are arranged in the following order:

Table C-11
DisplayCableCompCaps590 returned arrays

Range	100 kHz values	1 MHz values
2 e-12	2 pF low comp value	2 pF low comp value
2 e-12	2 pF high comp value	2 pF high comp value
20 e-12	20 pF low comp value	20 pF low comp value
20 e-12	20 pF high comp value	20 pF high comp value
200 e-12	200 pF low comp value	200 pF low comp value
200 e-12	200 pF high comp value	200 pF high comp value
2 e-9	2 nF low comp value	2 nF low comp value
2 e-9	2 nF high comp value	2 nF high comp value

Syntax:

status = DisplayCableCompCaps590(char *CabCompFile, double *Range, int RangeSize, double *Values100k, int Values100kSize, double *Values1M, int Values1MSize);

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

Values100kSize,(int)
Values1MSize,
RangeSize The size of the Range, Values100k, and Values1M arrays. THESE PARAMETERS MUST BE SET TO 8 TO AVOID UNPREDICTABLE RESULTS.

OUTPUTS:

Range (double array) An 8 element array that receives the nominal range values.

Values100 k (double array) An 8 element (fixed) array that receives the nominal capacitor values used to perform the cable compensation at the 100 kHz frequency.

Values1M (double array) An 8 element (fixed) array that receives the nominal capacitor values used to perform the cable compensation at the 1 MHz frequency.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0 OK.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

SaveCableCompCaps590 user module

Overview

This user module is used for Model 590 cable compensation. The user enters precise capacitance source values. When this test is run, the capacitance source values are saved to a user-specified file. The user module to perform cable compensation (CableCompensate590) can then access the capacitance source values from this file.

The default parameter values for this user module are shown in Figure C-25. These are suggested low / high values that can be used for cable compensation. You must replace these values with the calibration values of the capacitance sources.

Test example #1 demonstrates how cable compensation is performed (see Model 590 test examples earlier in this appendix).

Figure C-25
SaveCableCompCaps590 (default parameters)

Formulator

User Libraries: KI590ulib

User Modules: SaveCableCompCaps590

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\ki590ulib\\misc\\590cabcomp.
2	Lo2p100k	Input	DOUBLE	0.4e-12
3	Lo2p1M	Input	DOUBLE	0.4e-12
4	Hi2p100k	Input	DOUBLE	1.8e-12
5	Hi2p1M	Input	DOUBLE	1.8e-12
6	Lo20p100k	Input	DOUBLE	4e-12
7	Lo20p1M	Input	DOUBLE	4e-12
8	Hi20p100k	Input	DOUBLE	18e-12
9	Hi20p1M	Input	DOUBLE	18e-12
10	Lo200p100k	Input	DOUBLE	40e-12
11	Lo200p1M	Input	DOUBLE	40e-12
12	Hi200p100k	Input	DOUBLE	180e-12
13	Hi200p1M	Input	DOUBLE	180e-12
14	Lo2n100k	Input	DOUBLE	400e-12
15	Lo2n1M	Input	DOUBLE	400e-12
16	Hi2n100k	Input	DOUBLE	1800e-12
17	Hi2n1M	Input	DOUBLE	1800e-12

User module description

This function saves the nominal values of the capacitors used to perform the Model 590 cable compensation procedure to the indicated file. If no cable compensation file exists, then this module will create one provided that the user has the proper system permissions.

Syntax:

status = SaveCableCompCaps590(char *CabCompFile, double Lo2p100k, double Lo2p1M, double Hi2p100k, double Hi2p1M, double Lo20 p100 k, double Lo20 p1M, double Hi20 p100k, double Hi20 p1M, double Lo200 p100k, double Lo200 p1M, double Hi200 p100k, double 200 p1M, double Lo2n100k, double Lo2n1M, double Hi2n100k, double Lo2n1M);

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, use two backslash (\\) characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

Lo2p100 k (double) The nominal value of the low range capacitor used to perform cable compensation for the 2 pF range and 100 kHz frequency. The valid range of inputs is 0 to 0.95 e-12 farads.

Lo2p1M (double) The nominal value of the low range capacitor used to perform cable compensation for the 2 pF range and 1 MHz frequency. The valid range of inputs is 0 to 0.95e-12 farads.

Hi2p100k (double) The nominal value of the high range capacitor used to perform cable compensation for the 2 pF range and 100 kHz frequency. The valid range of inputs is 1 e-12 to 2 e-12 farads.

Hi2p1M (double) The nominal value of the high range capacitor used to perform cable compensation for the 2 pF range and 1 MHz frequency. The valid range of inputs is 1 e-12 to 2 e-12 farads.

Lo20p100k (double) The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency. The valid range of inputs is 0 to 9.5 e-12 farads.

Lo20p1M (double) The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency. The valid range of inputs is 0 to 9.5 e-12 farads.

Hi20p100k (double) The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency. The valid range of inputs is 10 e-12 to 20 e-12 farads.

Hi20p1M (double) The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency. The valid range of inputs is 10 e-12 to 20 e-12 farads.

Lo200p100k (double) The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency. The valid range of inputs is 0 to 95 e-12 farads.

Lo200p1M (double) The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency. The valid range of inputs is 0 to 95 e-12 farads.

Hi200p100k (double) The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency. The valid range of inputs is 100 e-12 to 200 e-12 farads.

Hi200p1M (double) The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency. The valid range of inputs is 100 e-12 to 200 e-12 farads.

Lo2n100k (double) The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency. The valid range of inputs is 0 to 995 e-12 farads.

Lo2n1M (double) The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency. The valid range of inputs is 0 to 995 e-12 farads.

Hi2n100k (double) The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency. The valid range of inputs is 1000 e-12 to 2000 e-12 farads.

Hi2n1M (double) The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency. The valid range of inputs is 1000 e-12 to 2000 e-12 farads.

OUTPUTS:

-**none**-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (the **Sheet** tab).

0 OK.

-**10000** (INVAL_INST_ID) An invalid instrument ID was specified. This generally means that there is no instrument with the specified ID in your configuration.

-**10001** (INVAL_PIN_SPEC) An invalid DUT pin number was specified.

-**10003** (NO_SWITCH_MATRIX) No switch matrix was found.

-**10004** (NO_MATRIX_CARDS) No matrix cards were found.

-**10020** (COMP_FILE_ACCESS_ERR) There was an error accessing the cable compensation file.

-**10021** (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-**10022** (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

Appendix D
Using an HP 4284A/4980A LCR Meter

In this section:

Key Concepts

NOTE: *For details on all aspects of HP Model 4284A operation, refer to the HP Model 4284A Operation Manual.*

NOTE: *For details on all aspects of Agilent Model 4980A operation, refer to the Agilent Model 4980A Operation Manual.*

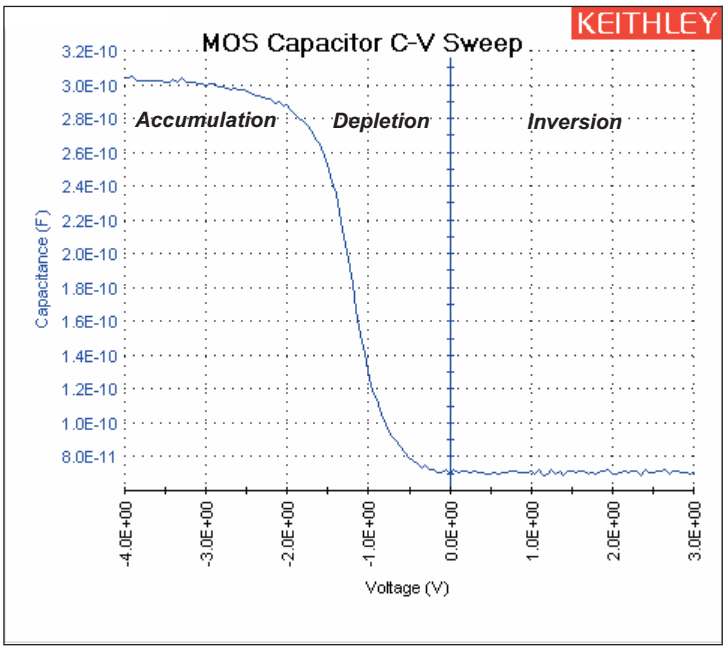
C-V measurement basics

The Keithley Instruments Model 4200-SCS can control the HP Model 4284A/4980A LCR Meter to measure capacitance versus voltage (C-V) of semiconductor devices. Typically, C-V measurements are performed on capacitor-like devices, such as a metal-oxide-silicon capacitor (MOS-C).

The purpose of MOS-C measurements is to study the integrity of the gate oxide and semiconductor doping profile, and the lifetime of semiconductor material. The interface quality between the gate oxide and silicon can also be studied. In addition, other dielectric materials used in an IC can be studied using the C-V technique.

A user-configured voltage sweep allows capacitance measurements that can span the three regions of a C-V curve; the accumulation region, depletion region, and inversion region. Figure D-1 shows the three regions of a typical C-V curve for a MOS-C.

Figure D-1
Typical C-V curve for a MOS-C



Capacitance measurement tests

The Model 4200-SCS provides the following user modules to perform C-V tests using the HP Model 4284A/4980A:

- **CvSweep4284: C-V sweep test**: Performs a capacitance and conductance measurement at each step of a user-configured linear voltage sweep.
- **Cmeas590: C measurement**: Performs a capacitance and conductance measurement at a fixed bias voltage.

Details on the user modules for the HP Model 4284A/4980A are provided in the hp4284ulib User Library Reference later in this section.

NOTE: *If desired, OPEN and SHORT correction can initially be performed on the Model 4284A/4980A to achieve the most accurate C-V measurements. See the HP Model 4284A/4980A Operation Manual for details.*

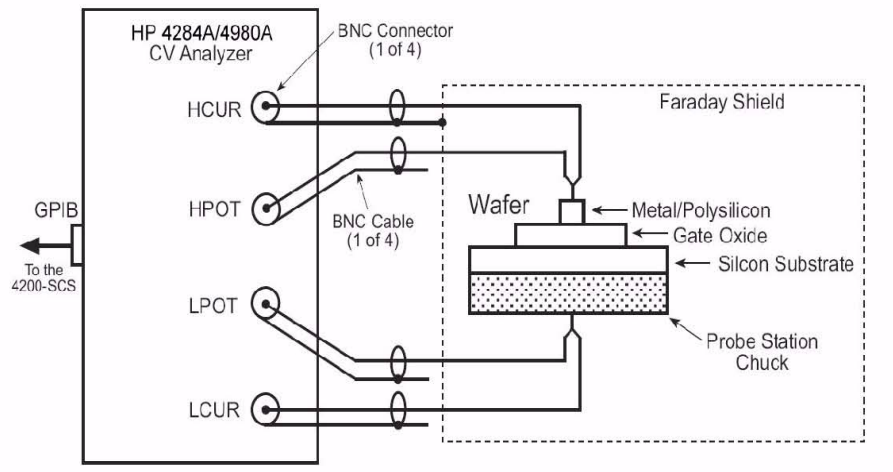
For details on Model 4284A/4980A connections, see the HP Model 4284A/4980A Operation Manual.

Connections

Signal connections

Basic 4-wire signal connections for the Model 4284A/4980A are shown in Figure D-2. The center conductors of the BNC connectors are connected to the DUT. The outer shield of one of the coaxial cables is typically connected to a Faraday shield. The Model 4284A/4980A output is typically connected to the wafer backside (or well). The input is typically connected to the gate of a MOS-C.

Figure D-2
Basic 4284A/4980A connections to DUT



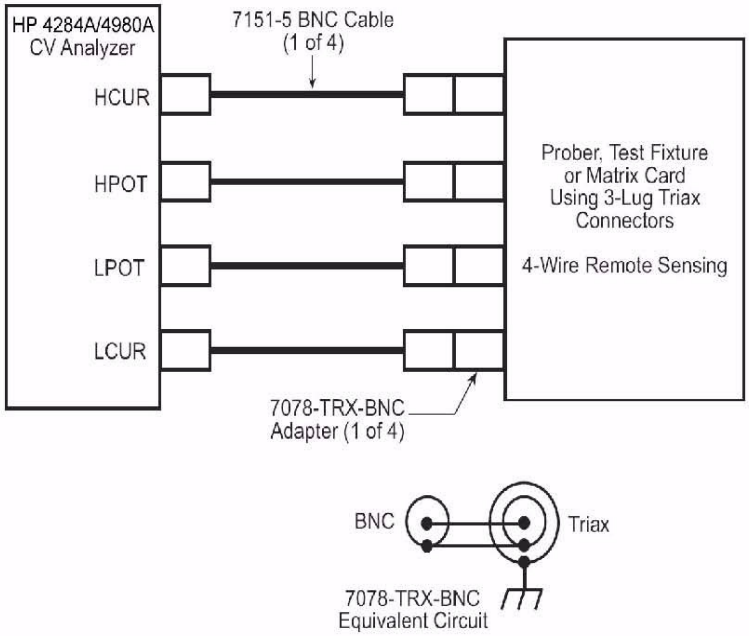
Triax connections

Adapters are required to connect the Model 4284A/4980A to equipment (for example, probe station, test fixture, matrix card) that uses triax connectors.

NOTE: *See Using Switch Matrices in Appendix B for details on using a switch matrix with the Model 4284A/4980A LCR Meter.*

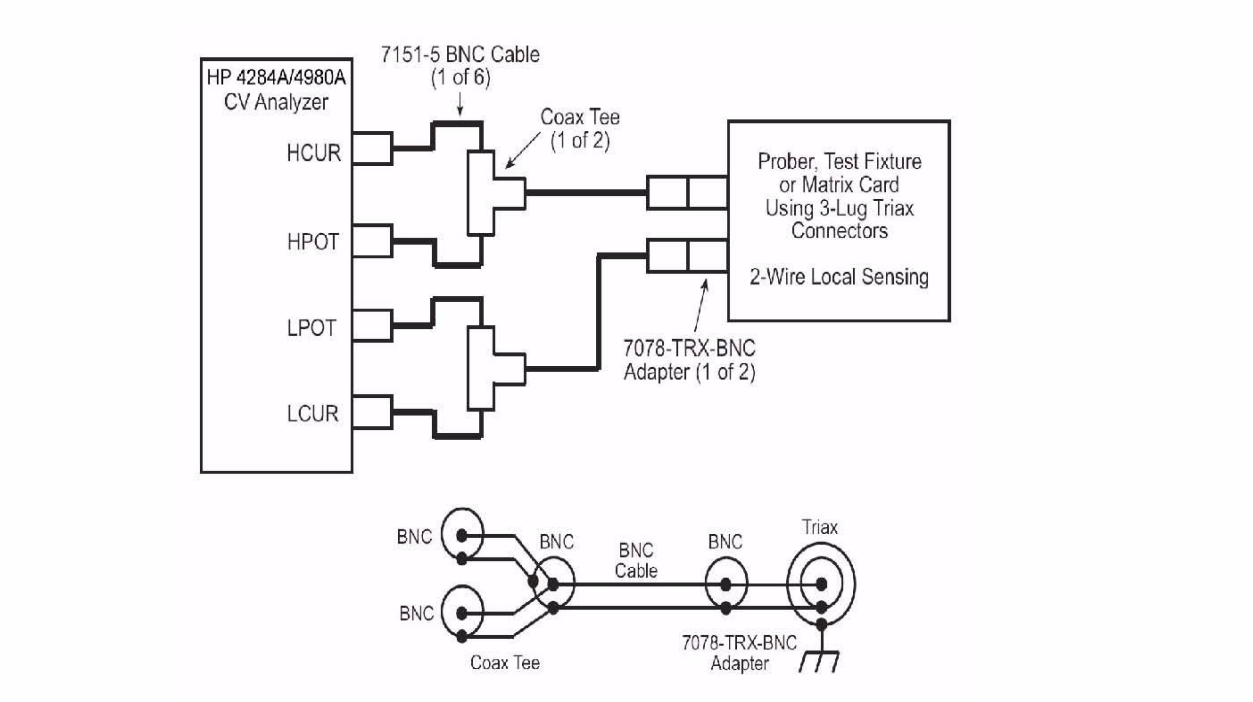
4-wire remote sensing: Figure D-3 shows 4-wire remote sense connections. The Model 7087-TRX-BNC is a 3-lug triax to BNC adapter. As shown in Figure D-3, connect the adapters to the 3-slot triax connectors, and then use a Model 7051-5 BNC cable to make the connections to the Model 4284A/4980A. Figure D-3 also shows the equivalent circuit for the adapter.

Figure D-3
4-wire remote sense connections to equipment using triax connectors



2-wire local sensing: For 2-wire local sense connections, Coax Tees are required to adapt dual-BNC to single-BNC, as shown in Figure D-4.

Figure D-4
2-wire local sense connections to equipment using triax connectors



GPIB Connections

The Model 4200-SCS controls the Model 4284A/4980A through the General Purpose Interface Bus (GPIB). Use the Model 7007-1 or 7007-2 GPIB cable to connect the GPIB port of the Model 4284A/4980A to the GPIB port of the Model 4200-SCS.

Using KCON to Add an Agilent (HP) LCR Meter to the System

In order for the Model 4200-SCS to control an external instrument, that instrument must be added to the system configuration. The Model 4284A/4980A is added to the test system using the KCON (Keithley CONfiguration utility) as follows:

Step 1. Close KITE and Open KCON

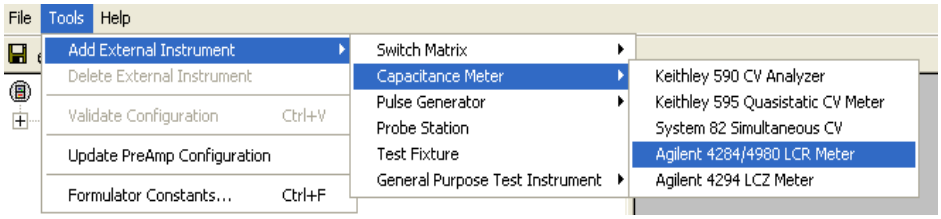
Close KITE by clicking the close button (**X**) at the top right-hand corner of the KITE panel. If you have made changes, you will be prompted to save them. On the windows desktop, double-click the KCON icon to open KCON.

For details on using KCON, see Keithley CONfiguration Utility (KCON) in Section 7.

Step 2. Add LCR Meter

Add the Agilent Model 4284A/4980A LCR Meter from the **Tools** menu on the Toolbar of the **KCON** window (Figure D-5). Figure D-6 shows the **Properties and Connections** window for the LCR Meter.

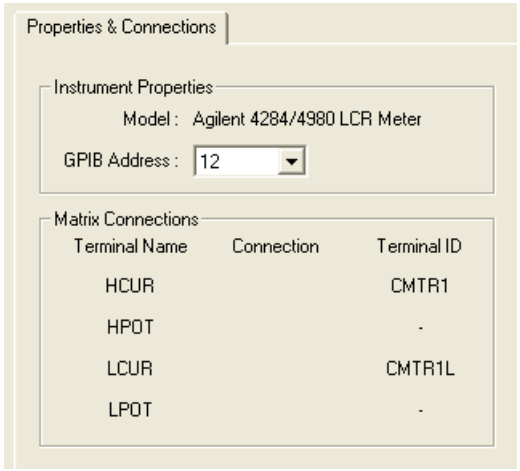
Figure D-5
KCON tools menu to add Model 4284A/4980A LCR Meter



Step 3. Set GPIB address

The GPIB address setting in the **Instrument Properties** area of the window (Figure D-6) must match the actual GPIB address of the Model 4284A/4980A.

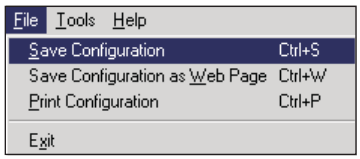
Figure D-6
4284A/4980A LCR Meter Properties & Connections Window



Step 4. Save Configuration

The KCON configuration is saved from the **File** menu on the toolbar. As shown in Figure D-7, click **Save Configuration**.

Figure D-7
Save KCON Configuration



Step 5. Close KCON and Open KITE

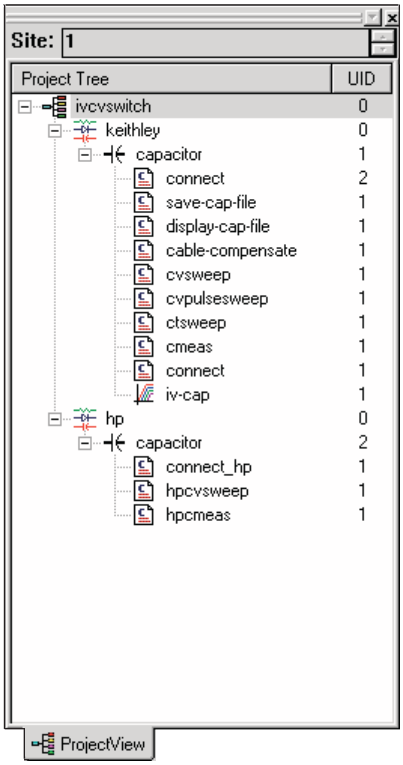
KCON can be closed from the **File** menu by clicking **Exit**. It can also by closed by clicking the close button (X) at the top right corner of the KCON window.

On the windows desktop, double-click the **KITE** icon to open KITE.

Model 4284A/4980A Test Example

The following test example for the Model 4284A/4980A LCR Meter is controlled by a UTM in the **ivcvswitch** project. Figure D-8 shows the Project Navigator for the project. A switch matrix is not used for this example.

Figure D-8
ivcvswitch Project Navigator



C-V sweep

This example assumes that the Model 4284A/4980A is already connected directly to the DUT. The DUT could be a device installed in a test fixture, or a MOS-C on a wafer. Perform the following steps to perform a C-V sweep.

Open and execute hpcvsweep UTM

1. In the Project Navigator (Figure D-8), double-click **hpcvsweep** to open the UTM (see Figure D-9). From the **Definition** tab, you can modify the test parameters, if desired. If you use the default parameters as shown in Figure D-9, the Model 4284A/4980A will perform a +3V to -4V staircase sweep using 50mV steps. A measurement will be performed on each step of the sweep.
2. Execute the test by clicking the green **Run** button.

hpcvsweep test description

As shown in Figure D-9, the **cvsweep** UTM uses the CvSweep4284 user module. For details on the test description, see the reference information for the CvSweep4284 user module in this appendix (hp4284ulib User Library Reference). Figure D-9 shows the complete default parameter list for the test, while Figure D-10 shows the configured sweep.

Figure D-1 shows a typical graph that is generated by this test.

Figure D-9
vSweep4284 user module (hpcvsweep UTM)

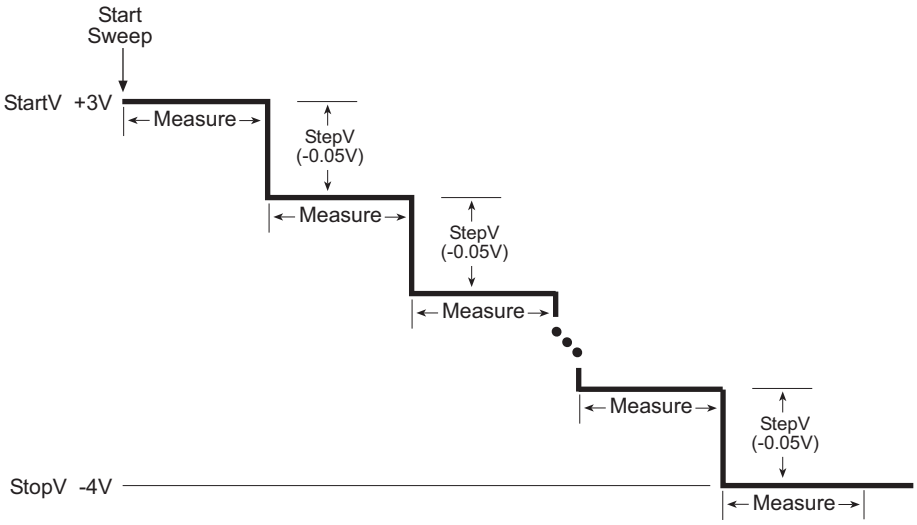
Formulator

User Libraries: HP4284ulib

User Modules: CvSweep4284

	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	CMTR1
2	LoPin	Input	INT	0
3	HiPin	Input	INT	0
4	StartV	Input	DOUBLE	3.000000e+000
5	StopV	Input	DOUBLE	-4.000000e+000
6	StepV	Input	DOUBLE	-5.000000e-002
7	Frequency	Input	DOUBLE	1.000000e+005
8	Range	Input	DOUBLE	1.000000e+002
9	Model	Input	INT	1
10	IntegrationTime	Input	INT	1
11	C	Output	DBL_ARRAY	
12	Csize	Input	INT	141
13	V	Output	DBL_ARRAY	
14	Vsize	Input	INT	141
15	G_or_R	Output	DBL_ARRAY	
16	G_or_Rsize	Input	INT	141

Figure D-10
C-V linear staircase sweep



hp4284ulib User Library Reference

The user modules in the hp4284ulib user library are used to control the Model 4284A/4980A LCR Meter. These user modules are summarized in Table D-1. Also listed in the table are the Keithley Instruments-created UTM names that use the user modules.

Details for each of the user modules follow the table.

Table D-1
hp4284ulib User Library

User module	UTM name	Description
CvSweep4284	hpcvsweep	Performs capacitance vs. voltage measurements using a staircase sweep.
Cmeas4284	hpcmeas	Performs a single capacitance measurement.

CvSweep4284 User Module

Overview

This user module performs a capacitance versus voltage staircase sweep. Figure D-9 shows the default parameters for the **hpcvsweep** UTM which uses the CvSweep4284 user module. In this example, the Model 4284A/4980A outputs a linear staircase voltage sweep from +3V to -4V in 50mV steps. As shown in Figure D-10, a capacitance measurement is performed on each step of the sweep. A test example demonstrates how to perform a C-V sweep (see Model 4284A/4980A Test Example earlier in this section).

The user-entered parameters are explained in User Module Description in the following text.

User Module Description

The CvSweep4284 routine performs a capacitance vs. voltage (C-V) sweep using the Agilent (HP) Model 4284A/4980 LCR Meter.

Syntax:

status = CvSweep4284(char *InstIdStr, int LoPin, int HiPin, double StartV, double StopV, double StepV, double SignalLevel, double Frequency, double Range, int Model, int IntegrationTime, double *C, int Csize, double *V, int Vsize, double *G_or_R, int G_or_Rsize);

INPUTS:

InstIdStr (char *) The CMTR instrument ID. This will be either CMTR1 or CMTR2, dependent upon your system's configuration.

LoPin (int) The DUT pin which the 4284A/4980's low terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the low terminal will be connected to the specified pin. Valid values for this parameter are -1 to 72.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, "**connect**"), there is no need to connect LoPin and HiPin. Set these parameters to 0.*

HiPin (int) The DUT pin which the 4284A/4980's high terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the high terminal will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the previous NOTE.

StartV (double) The starting voltage of the sweep. The valid range of input values is -40 to +40 volts.

StopV (double) The ending voltage of the sweep. The valid range of input values is -40 to +40 volts.

StepV	(double) The sweep voltage step size. The valid range of input values is -40 to +40 volts. The value of ((StopV – StartV)/StepV) + 1 must be less than or equal to the values for Csize, Vsize, and G_or_Rsize.
SignalLevel	(double) The oscillator output voltage level. The valid range of values is 5E-3 through 20 volts.
Frequency	(double) A variable that selects the measurement frequency to use. Valid inputs are 20 through 1e6 Hertz.
Range	(double) The measurement range to use. Valid values for this parameter are 0 (Auto), 100, 300, 1000, 3000, 10000, 30000, and 100000 Ohms.
Model	(int) Which measurement model to use. Entering 0 for this parameter will select the series model, while 1 will select the parallel model.
IntegrationTime	(int) The integration time to use. Valid values are: 0 (SHORT), 1 (MEDIUM), and 2 (LONG).
Csize, Vsize, G_or_Rsize	(int) These parameters must be set to a value equal to or greater than the number of steps in the sweep or, = ((StopV - StartV)/StepV) + 1. When this function is called from a KITE UTM, these values are fixed at 1350.

OUTPUTS:

C	(double *) The measured array of capacitance values.
V	(double *) The array of voltage biases used in the sweep.
G_or_R	(double *) If the parallel Model (1) is selected, G_or_R is the measured conductance. For the series Model (0), G_or_R is the measured resistance.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0	OK.
-10030	(HP4284_NOT_IN_KCON) No HP4284A/4980 LCR is defined in your system's configuration.
-10031	(HP4284_MEAS_ERROR) A measurement error occurred.
-10090	(GPIB_ERROR_OCCURRED) A GPIB communications error occurred.
-10091	(GPIB_TIMEOUT) A time-out occurred during communications.
-10100	(INVAL_PARAM) An invalid input parameter was specified.
-10102	(ERROR_PARSING) There was an error parsing the 4284A/4980's response.
-10101	(ARRAY_SIZE_TOO_SMALL) The specified value for Csize, G_or_Rsize, Vsize, or Tsize was too small for the number of steps in the sweep.

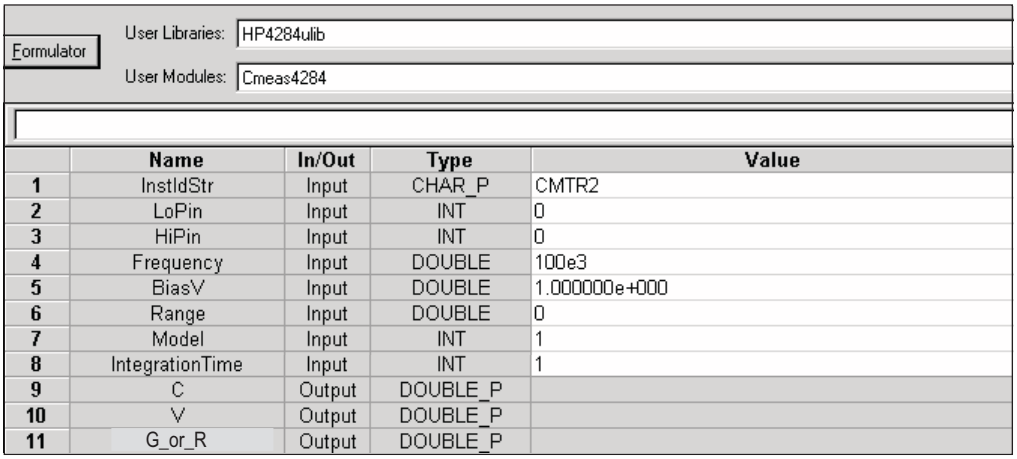
Cmeas4284 User Module

Overview

This user module is used to perform a single, fixed-bias capacitance and conductance measurement. Figure D-11 shows the default parameters for the **hpcmeas** UTM which uses the Cmeas4284 user module. In this example, the Model 4284A/4980A is set to source 1V, and a capacitance measurement is performed.

The user-entered parameters are explained in User Module Description in the following text.

Figure D-11
meas4284 (hpcmeas UTM)



	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	CMTR2
2	LoPin	Input	INT	0
3	HiPin	Input	INT	0
4	Frequency	Input	DOUBLE	100e3
5	BiasV	Input	DOUBLE	1.000000e+000
6	Range	Input	DOUBLE	0
7	Model	Input	INT	1
8	IntegrationTime	Input	INT	1
9	C	Output	DOUBLE_P	
10	V	Output	DOUBLE_P	
11	G_or_R	Output	DOUBLE_P	

User Module Description

The Cmeas4284 routine measures capacitance and conductance using the Agilent (HP) Model 4284A/4980 LCR Meter.

Syntax:

status = Cmeas4284(char *InstIdStr, int LoPin, int HiPin, double Frequency, double BiasV, double Range, int Model, int IntegrationTime, double *C, int Csize, double *V, int Vsize, double *G_or_R, int G_or_Rsize);

INPUTS:

InstIdStr (char *) The CMTR instrument ID. This will be either CMTR1 or CMTR2, dependent upon your system's configuration.

LoPin (int) The DUT pin which the 4284A/4980's low terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the low terminal will be connected to the specified pin. Valid values for this parameter are -1 to 72.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, "**connect**"), there is no need to connect LoPin and HiPin. Set these parameters to 0.*

HiPin (int) The DUT pin which the 4284A/4980's high terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the high terminal will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the previous NOTE.

Frequency (double) A variable that selects the measurement frequency to use. Valid inputs are 20 through 1e6 Hertz.

BiasV (double) The DC bias to use for the measurement. Valid inputs are -40 to +40 volts.

Range (double) The measurement range to use. Valid values for this parameter are 0 (Auto), 100, 300, 1000, 3000, 10,000, 30,000, and 100,000 Ohms.

Model (int) Which measurement model to use. Entering 0 for this parameter will select the series model, while 1 will select the parallel model.

IntegrationTime (int) The integration time to use. Valid values are: 0 (SHORT), 1 (MEDIUM), and 2 (LONG).

OUTPUTS:

C (double *) The measured capacitance.

V (double *) The bias voltage used.

G_or_R (double *) If the parallel Model (1) is selected, G_or_R is the measured conductance. For the series Model (0), G_or_R is the measured resistance.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0 OK.

-10000 (INVAL_INST_ID) The specified instrument does not exist.

-10030 (HP4284_NOT_IN_KCON) No HP4284A/4980 LCR is defined in your system's configuration.

-10031 (HP4284_MEAS_ERROR) A measurement error occurred.

-10090 (GPIB_ERROR_OCCURRED) A GPIB communications error occurred.

-10091 (GPIB_TIMEOUT) A time-out occurred during communications.

-10102 (ERROR_PARSING) There was an error parsing the 4284A/4980's response.

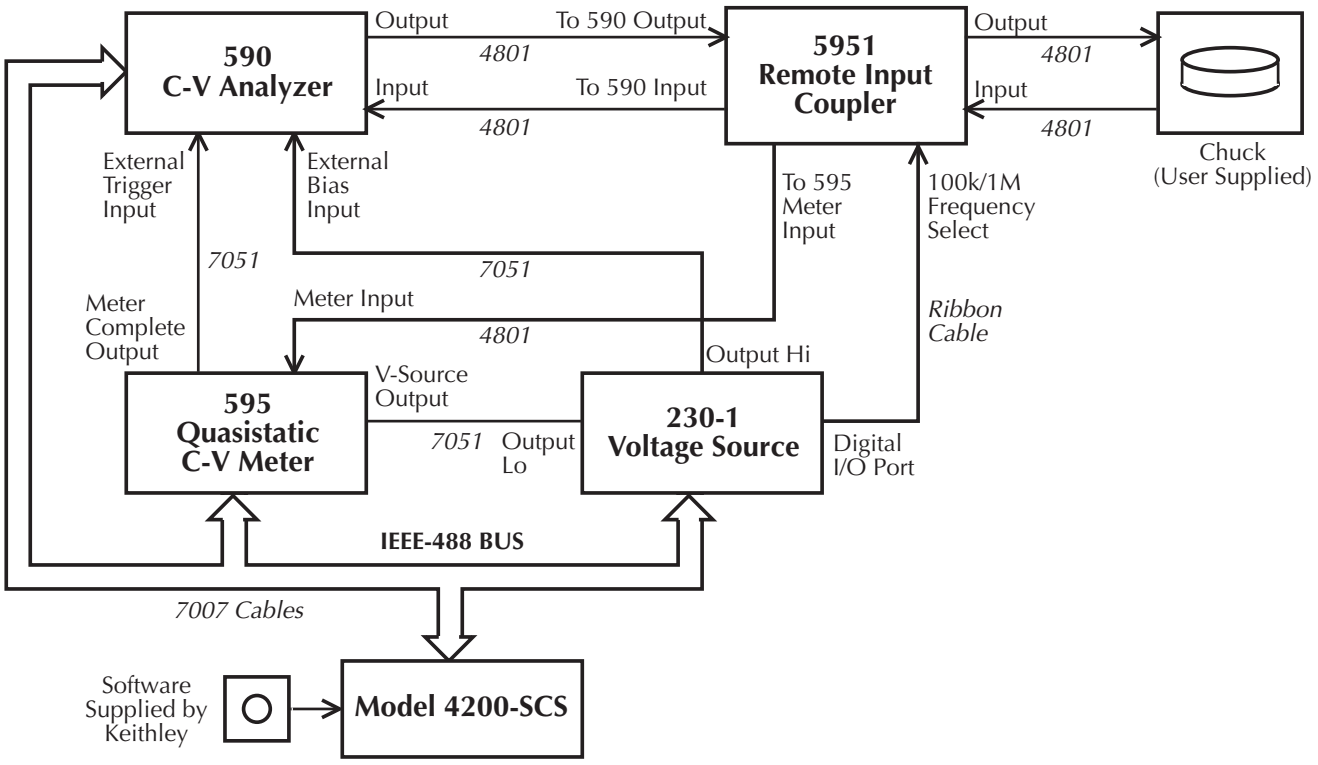
Appendix E
Using a Keithley Model 82 C-V System

In this section:

Key concepts

The Model 82 C-V System uses a Keithley Instruments Model 590 C-V Analyzer and a Keithley Instruments Model 595 Quasistatic C-V Meter to perform simultaneous C-V measurements. The complete system is shown in Figure E-1. Project plans for the Model 4200-SCS are provided to perform simultaneous C-V measurements, STVS measurements for mobile ion extraction and minority carrier generation lifetime measurements.

Figure E-1
System block diagram



Capacitance measurement tests

The Model 4200-SCS provides the following user modules to perform capacitance tests using the Model 82:

- **CtSweep82: C-t measurements**: Performs a specified number of capacitance. measurements at a specified time interval. Voltage is held constant for these capacitance measurements.
- **SIMCVsweep82: Simultaneous C-V sweep test**: Performs a simultaneous capacitance vs. voltage (C-V) sweep.
- **QTsweep82: Quasistatic capacitance and leakage current test**: Measures quasistatic capacitance and leakage current, as a function of delay time, to determine the equilibrium condition.

NOTE: *Details on all user modules for the Model 82 are provided in "ki82ulib user library reference" later in this appendix.*

C-t measurements

A C-t sweep performs a specified number of capacitance measurements at a specified time interval with voltage held constant. Figure E-2 shows an example of a C-t waveform.

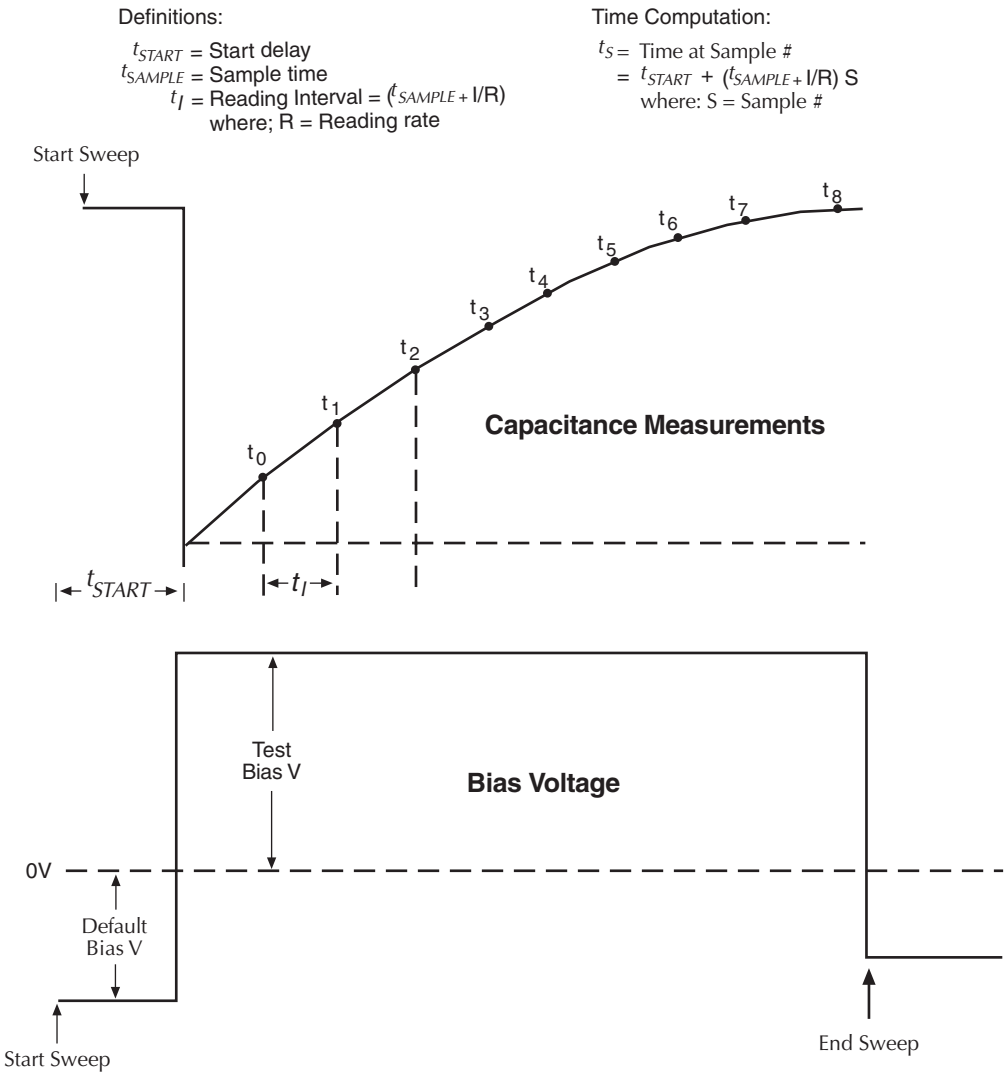
When the sweep is started, the device is stressed at a default voltage for a specified period of time. The test bias is then applied and a specified number of capacitance measurements are performed at a specific time interval.

The time interval between each reading is the sum of the specified time between samples (Sample_Time) and the reading rate time (as determined by Reading_Rate) for each measurement.

NOTE: *See Model 82 project plans for details on the test to perform C-t measurements.*

Details on all parameters for the test using the CtSweep82 user module are provided in the ki82ulib user library reference later in this appendix (see User module description).

Figure E-2
C-t waveform



Simultaneous C-V measurements

For simultaneous C-V measurements, the Models 590 and 595 both measure capacitance during the same voltage sweep. The readings from the two instruments are synchronized using external triggering and are taken alternately during the sweep.

Figure E-3 shows a simplified representation of the stepped bias voltage supplied by the Model 595 during a measurement sweep. Each vertical voltage step size depends on the programmed Model 595 bias step, while each horizontal time step is determined by the programmed delay time.

A quasistatic measurement is a two-step process requiring at least two charge measurements. Initially, at the end of step S_1, the first charge measurement Q_1 is made, after which the voltage goes to the next step. Following the programmed delay period, the Q_3 charge measurement is made, and the capacitance is then calculated from these values and the step size. Here we see that two voltage steps are necessary for every low-frequency capacitance measurement.

The Model 590 is triggered one delay time after the completion of each Model 595 reading. As a result, high-frequency measurements are made on only every other step (as represented by the small rectangles in Figure E-3). Furthermore, notice that the high-frequency measurements are not made at exactly the same voltage as the quasistatic measurements. High frequency

capacitance measurements CH_m and CH_{m+1} are made at voltages VH_m and VH_{m+1}, respectively. Quasistatic measurements by their very nature result from the charge transfer as the voltage transitions from one step to the next, so that quasistatic capacitance measurement CQ_m is reported at a voltage half-way between the voltages at which its charge measurements Q_1 and Q_3 are made, which is $VQ_m = (V_n\text{-}0.5*V_{step})$.

To compensate for this voltage skew, an adjusted quasistatic capacitance value is calculated by interpolation to correspond to the voltages at which the high frequency measurements were made. The result is a new array of capacitance values CQ'_n corresponding to each high frequency result, CH_n and VH_n.

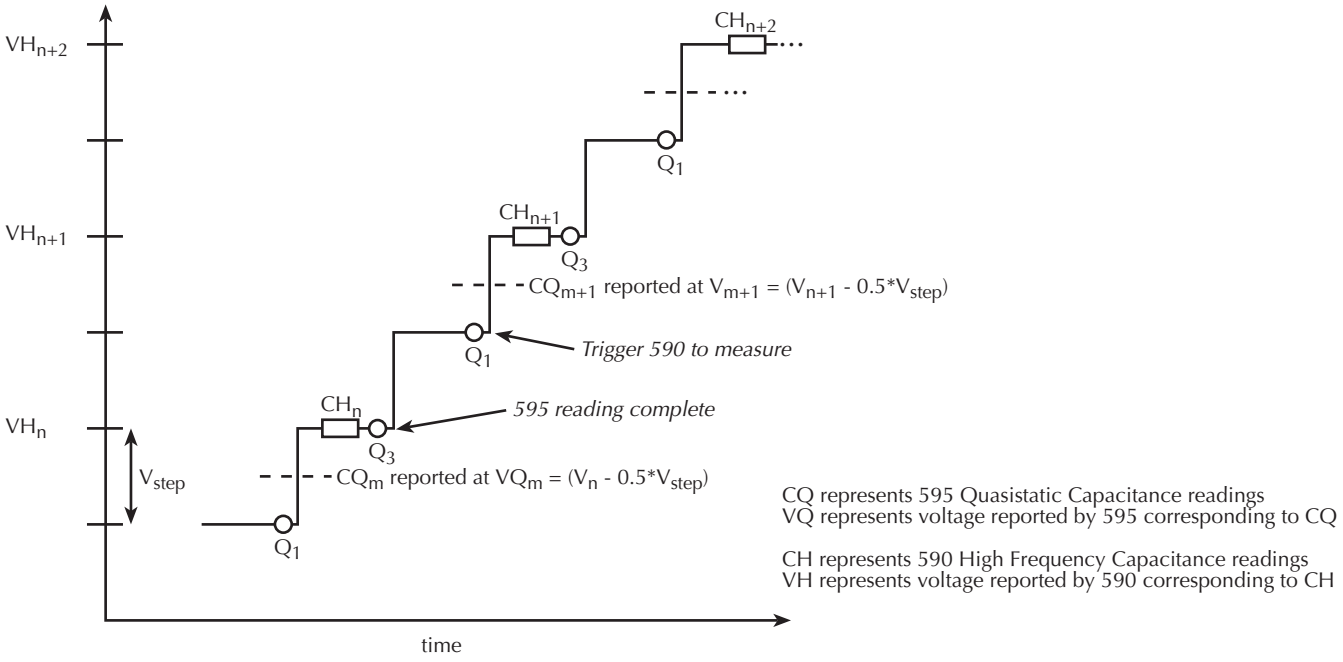
$$CQ'_n = CQ(VH_n) = CQ_m + [(CQ_{m+1} - CQ_m) / (VQ_{m+1} - VQ_m)] * (V_{step}/2) = CQ_m + [(CQ_{m+1} - CQ_m) / 4]$$

NOTE: *See "Model 82 project plans" for details on the test to perform simultaneous C-V measurements.*

Details on all parameters for the test are provided in the "ki82ulib user library reference" later in this appendix for the CVsweep82 user module (see "User module description").

NOTE: *As shown in Figure E-3, the first high frequency measurement (CH1) is performed during the second phase of the voltage sweep. Only quasistatic capacitance (C_1) will be measured during the first phase and will be disregarded.*

Figure E-3
Simultaneous C-V waveform



Cable compensation

Ideally, the Model 82 would only measure the capacitance of the DUT. However, signal pathways through the test cables, switch matrix, test fixture, and prober contribute unwanted capacitances that may adversely affect the measurement.

To correct for these unwanted capacitances, cable compensation should be performed before measuring the capacitance of DUT. In general, cable compensation is performed by connecting

precisely known capacitance sources in place of the DUT and then measuring them. The Model 590 then uses these measured values to perform correction when measuring the DUT.

Cable compensation involves two steps:

1. The Model 82 calculates the compensation parameters based on the comparison between the given and measured values.
2. The Model 82 performs a probe-up offset measurement and suppresses any remaining offset capacitance. This step is performed every time a new measurement is performed.

Typically, cable compensation is performed for all four measurement ranges (2pF, 20 pF, 200 pF, and 2 nF) of the Model 590. Once cable compensation is performed, it does not have to be done again unless the connection scheme to the DUT is changed, or power is cycled.

For each measurement range of the Model 590, a low capacitance source and a high capacitance source must be used. The Model 5906 Calibration Sources has the capacitance sources that can be used for cable compensation. Table E-4 lists the Model 5906 capacitance sources that can be used for each Model 590 range.

Table E-1
Model 5906 capacitance sources

590 range	Low capacitance source	High capacitance source
2pF	0.5 pF	1.5 pF
20 pF	4.7pF	18 pF
200 pF	47pF	180 pF
2 nF	470 pF	1.8nF

Cable compensation tests

The Model 82 has three user modules associated with cable compensation:

- **SaveCableCompCaps82: Enter and save capacitance source values**: The user enters the actual capacitance values of the capacitance sources. When the test is executed, the capacitance values are stored in a file at a user-specified directory path.
- **DisplayCableCompCaps82: Places capacitance values into a spread sheet**: When this test is executed, the capacitance values saved by SaveCableCompCaps82 are placed into its **Sheet** tab.
- **CableCompensate82: Performs cable compensation**: The user specifies the ranges and test frequencies for cable compensation. When this test is executed, on-screen prompts will guide you through the cable compensation process.

CabCompFile: Each of the above three user modules for cable compensation use a cable compensation file to save / load capacitor source values. Therefore, all three user modules must use the same file directory path.

Connections

Figure E-1 shows the overall system configuration for the Model 82. Connect all cables as shown in the diagram.

Figure E-4 shows how to connect the Model 5951 Remote Input Coupler to the Model 590. Take one low noise Model 4801 BNC cable and connect the 590 INPUT on the front of the Model 590 to the TO 590 INPUT on the back of the Model 5951. Use another Model 4801 cable and connect the 590 OUTPUT, also on the front of the Model 590, to the TO 590 OUTPUT on the back of the Model 5951. Connect two more low noise cables to the front of the Model 5951, where the input and output to the device are located. Connect the dark box to the cable grounds only. If this is not possible, connect a #18 AWG wire between the dark box and the white banana jack on the back of the Model 595.

Figure E-4
System front panel connections

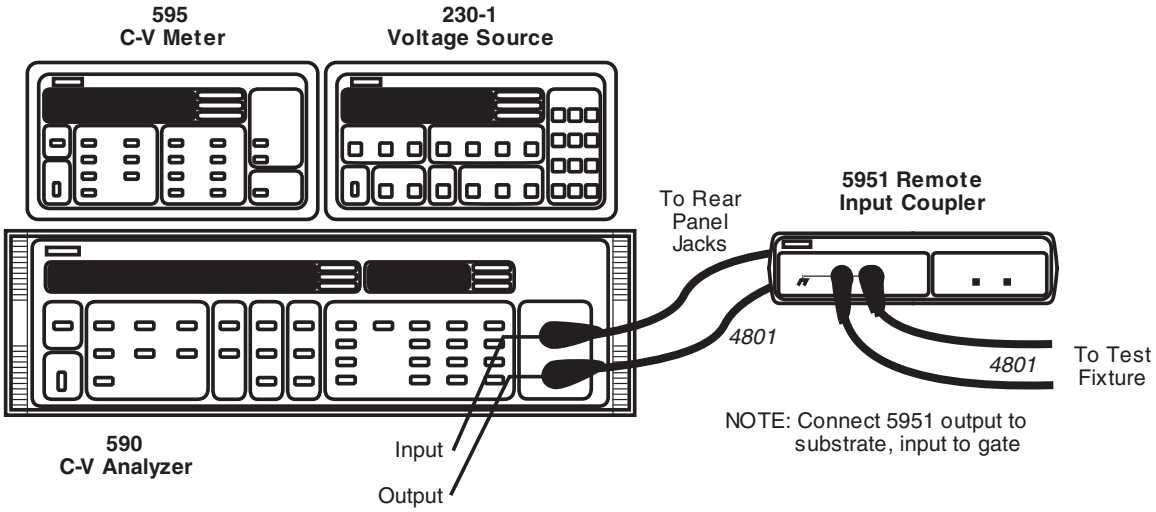
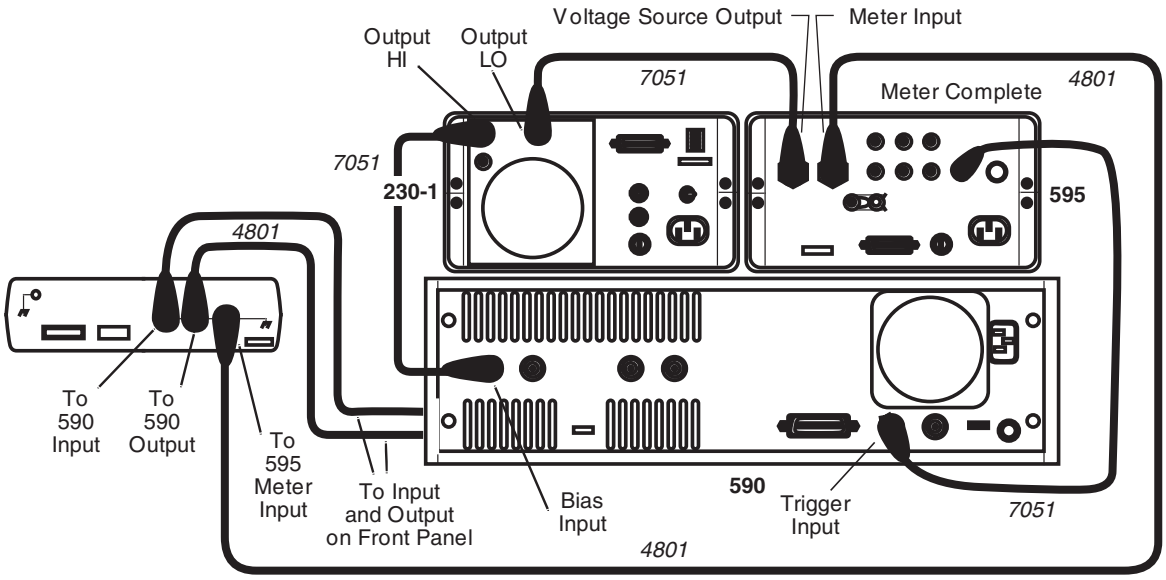


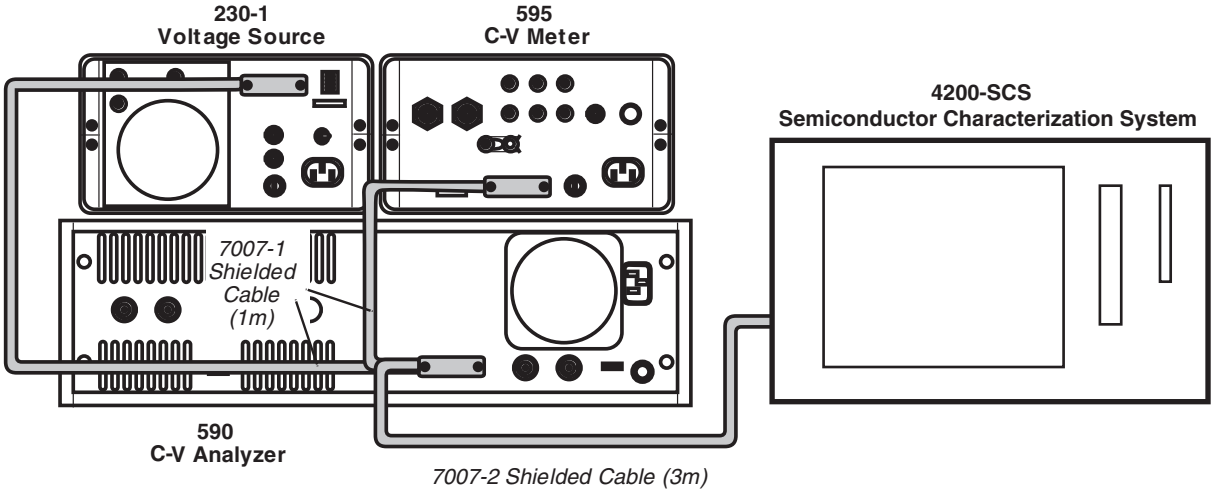
Figure E-5 shows the rest of the main cabling configuration. Take the final Model 4801 cable and connect the METER INPUT on the back of the Model 595 to the TO 595 INPUT on the Model 5951. Take a Model 7051-2 BNC cable and connect the METER COMPLETE port on the back of the Model 595 to the TRIGGER INPUT on the back of the Model 590. Take another Model 7051-2 cable and connect the OUTPUT HI on the back of the Model 230-1 to the BIAS INPUT on the back of the Model 590. Use the remaining Model 7051-2 BNC cable to connect the OUTPUT LO on the back of the Model 230-1 to the VOLTAGE SOURCE OUTPUT on the back of the Model 595.

Figure E-5
System rear panel connections



Next, attach the power cords to the devices. Use the ribbon cable to connect the DIGITAL I / O PORT on the back of the Model 230-1 to the TO 230-1 DIGITAL I / O on the back of the Model 5951. Take the power cables and plug in the units. Figure E-6 shows how to connect the IEEE-488 bus cables. Take the IEEE-488 bus cables and connect the Model 590, the Model 595, and the Model 230-1 to the Model 4200-SCS through the IEEE-488 card.

Figure E-6
System IEEE-488 connections



Using KCON to add Model 82 C-V System

In order for the Model 4200-SCS to control instruments in the C-V system, that system must be added to the Model 4200-SCS system configuration. The Model 82 C-V System is added to the test system using the KCON (Keithley CONfiguration utility) as follows:

Step 1. Close KITE and open KCON

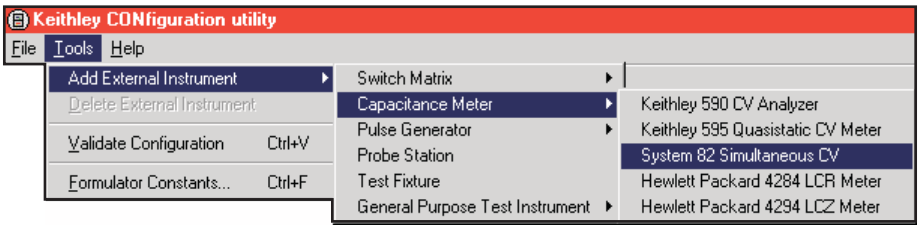
Close KITE by clicking the close button (**X**) at the top right-hand corner of the KITE panel. If you have made changes, you will be prompted to save them. On the windows desktop, double-click the **KCON** icon to open KCON.

For details on using KCON, refer to Section 7, Keithley CONfiguration Utility (KCON).

Step 2. Add a Model 82 C-V System

Add the Keithley Instruments Model 82 C-V System from the **Tools** menu on the Toolbar of the KCON window (Figure E-7). Figure E-8 shows the Keithley Instruments Model 82 C-V System added to the system.

Figure E-7
KCON tools menu to add Model 82 C-V System

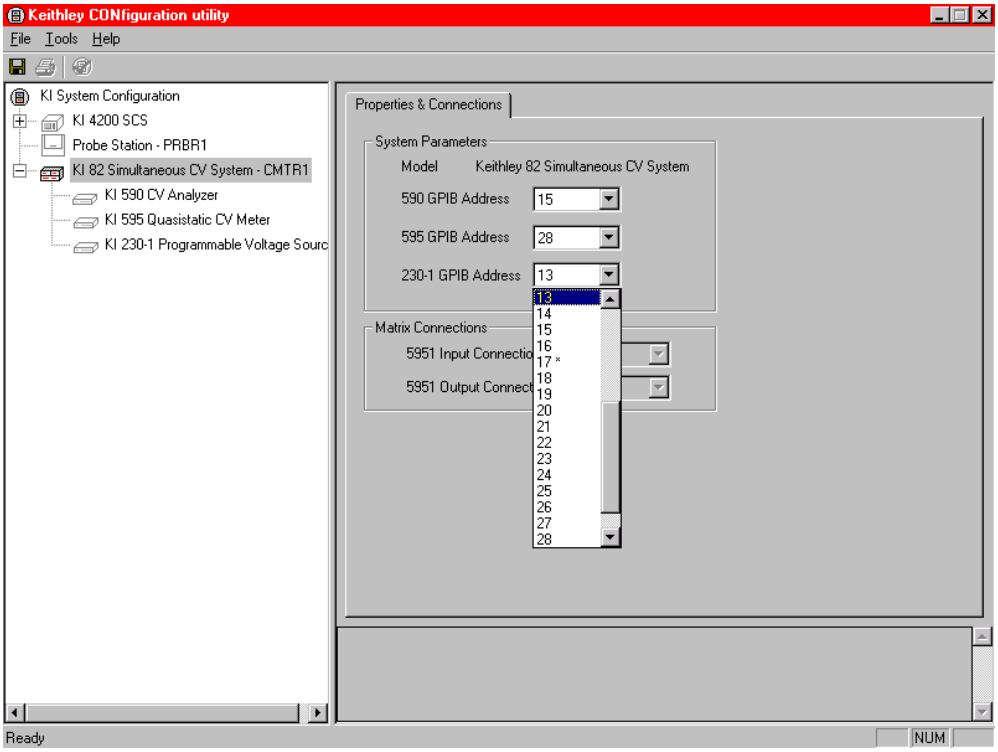


Step 3. Set GPIB addresses

The GPIB address settings in the **System Parameters** area of the window (Figure E-8) must match the actual GPIB address of the instruments in the system. The address for each instrument in the C-V system is briefly displayed during its power-on sequence.

Each instrument must have its own unique address value. As shown in Figure E-8, an address value can be changed from a drop-down menu. Note that the address value with the asterisk (*) is the GPIB address of the Model 4200-SCS. Do not use this address setting for any of the instruments in the C-V system.

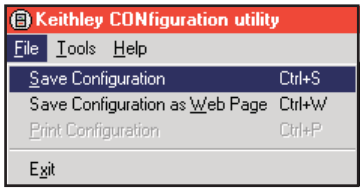
Figure E-8
Model 82 C-V System added to Model 4200-SCS System



Step 4. Save configuration

The KCON configuration is saved from the **File** menu on the toolbar. As shown in Figure E-9, click **Save Configuration**.

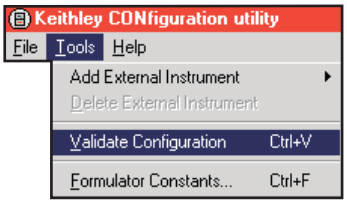
Figure E-9
Save KCON configuration



Step 5. Validate configuration

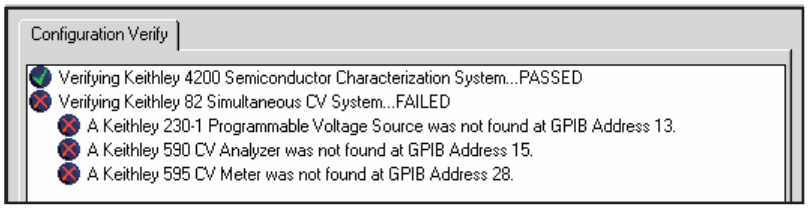
Validation checks communications between the Model 4200-SCS and the individual instruments in the C-V system. As shown in Figure E-10, validation is performed by selecting the **Validate Configuration** item of the **Tools** menu.

Figure E-10
Validate configuration



A window appears to report the results of the validation checks. Figure E-11 shows an example of a report for failed validation. In the event of a failure, ensure the instrument is turned on, it is properly connected to the GPIB, and its address matches its address setting in KCON.

Figure E-11
Example report of validation errors



Step 6. Close KCON and open KITE

KCON can be closed from the **File** menu by clicking **Exit**. It can also by closed by clicking the **close** button (X) at the top right-hand corner of the KCON window.

On the windows desktop, double-click the **KITE** icon to open KITE.

Model 82 project plans

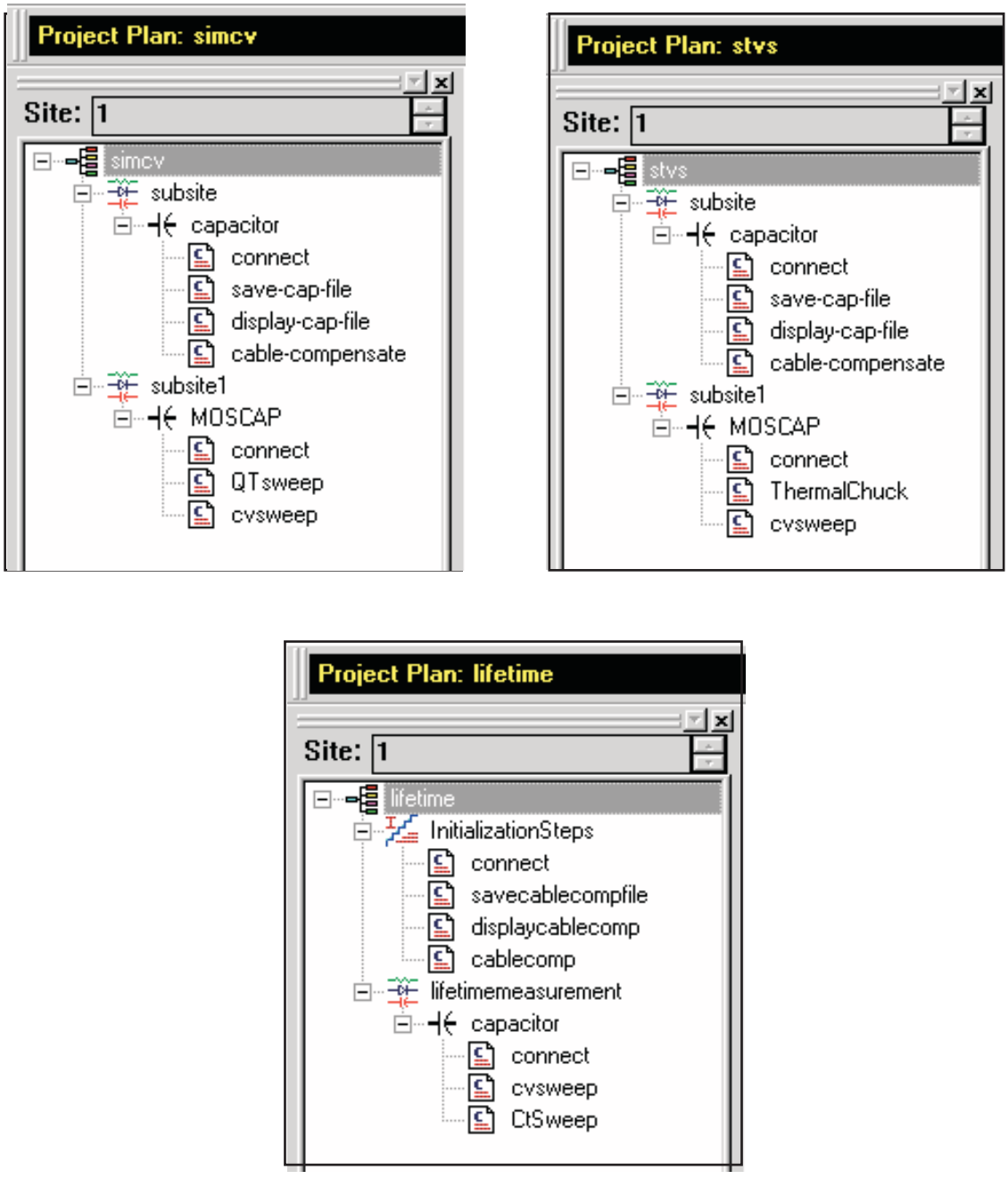
There are three project plans for the Model 82; simcv, stvs, and lifetime. The Project Navigators for these projects are shown in Figure E-12. Each project begins by performing tests for cable compensation. After cable compensation, each project then performs one or more capacitance tests.

- **simcv**: This project first uses the **QTsweep** test (QTsweep82 user module) to perform quasistatic capacitance measurements. This test is used to optimize delay time for quasistatic measurements so that the entire simultaneous C-V test is performed at DUT equilibrium. Then, **cvsweep** test (SIMCVsweep82 user module) is used to perform simultaneous C-V measurements.
- **stvs**: This project uses the **ThermalChuck** test to prompt the user to increase the temperature of the thermal chuck, and then uses the **cvsweep** test (SIMCVsweep82 user module) to perform simultaneous C-V measurements.
- **lifetime**: This project uses the **cvsweep** test (**SIMCVsweep82** user module) to perform simultaneous C-V measurements, and then uses the **CtSweep** test (CTsweep82 user module) to perform C-t measurements at the condition determined by the **cvsweep** test.

The user modules to perform cable compensation and capacitance tests are explained as follows.

NOTE: *Details on all parameters for the compensation and capacitance tests are provided in the "ki82ulib user library reference" later in this appendix (see the appropriate "User module description").*

Figure E-12
Model 82 project plans



Cable compensation tests

These tests assume that the calibration capacitors are installed as close to the wafer chuck-end of the cable as possible. Perform the following steps to perform cable compensation.

NOTE: *The three user modules for cable compensation must share the same file for capacitance source values. Therefore, the same file directory path must be used in all three user modules. For this example, use the default file directory path (see line 1 of the parameter list in Figure E-12, Figure E-13, and Figure E-15).*

A. Enter and save capacitance source values (SaveCableCompCaps82)

1. Depending on which project you're using (Figure E-12), double-click **save-cap-file** or **savecablecompfile** to open the UTM. These UTMs use the **SaveCableCompCaps82** user module. Figure E-13 shows the user module.
2. In the parameter list, enter the capacitance source calibration value for each range and frequency. If using the Model 5906, each capacitor has a label indicating the calibration value at 100 kHz and at 1 MHz.

   For example, assume the low capacitance source for the 2pA range is 0.47773 pF (100 kHz) and 0.47786 pA (1 MHz). Enter these values using scientific notation:

   Line 2 (Lo2p100 k) - Enter 0.47773e-12
   Line 3 (Lo2p1M) - Enter 0.47786e-12
3. In the Project Navigator, click **save-cap-file** or **savecablecompfile** to select the UTM, and then click the green run key to execute the test. The capacitor source values entered into the UTM will be saved in the file using the directory path specified in line 1 of the parameter list.

Figure E-13
SaveCableCompCaps82 user module

Formulator
User Libraries: KI82ulib
User Modules: SaveCableCompCaps82

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	Lo2p100k	Input	DOUBLE	4.294300e-013
3	Lo2p1M	Input	DOUBLE	4.294300e-013
4	Hi2p100k	Input	DOUBLE	1.292700e-012
5	Hi2p1M	Input	DOUBLE	1.292700e-012
6	Lo20p100k	Input	DOUBLE	4.862500e-012
7	Lo20p1M	Input	DOUBLE	4.862500e-012
8	Hi20p100k	Input	DOUBLE	1.759200e-011
9	Hi20p1M	Input	DOUBLE	1.759300e-011
10	Lo200p100k	Input	DOUBLE	4.779500e-011
11	Lo200p1M	Input	DOUBLE	4.779900e-011
12	Hi200p100k	Input	DOUBLE	1.779200e-010
13	Hi200p1M	Input	DOUBLE	1.779200e-010
14	Lo2n100k	Input	DOUBLE	4.626300e-010
15	Lo2n1M	Input	DOUBLE	4.630400e-010
16	Hi2n100k	Input	DOUBLE	1.766900e-009
17	Hi2n1M	Input	DOUBLE	1.772600e-009

B. Place capacitance source values in a spreadsheet (DisplayCableCompCaps82)

1. In the Project Navigator (Figure E-12), double-click **display-cap-file** or **displaycablecomp** to open the UTM. Figure E-14 shows the parameter list for the **DisplayCableCompCaps82** user module.
2. Ensure that line 1 of the parameter list has the same file directory path that is used in Step 1 (Figure E-13). Lines 3, 5, and 8 set array size. These must be set to 8 as shown in Figure E-14.
3. Execute the **display-cap-file** UTM by clicking the green run button. The calibration source values entered and saved in step A are placed into its spread sheet.
4. In the Workspace, click the **Sheet** tab for **display-cap-file** to display its spreadsheet. An example spreadsheet is shown in Figure E-15.

Figure E-14
DisplayCableCompCaps82 user module

Formulator

User Libraries: KI82ulib

User Modules: DisplayCableCompCaps82

c:\\S4200\\kiuser\\usrlib\\KI82ulib\\KI82cabcomp.dat

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	Range	Output	DBL_ARRAY	
3	RangeSize	Input	INT	8
4	Values100k	Output	DBL_ARRAY	
5	Values100kSize	Input	INT	8
6	Values1M	Output	DBL_ARRAY	
7	Values1MSize	Input	INT	8

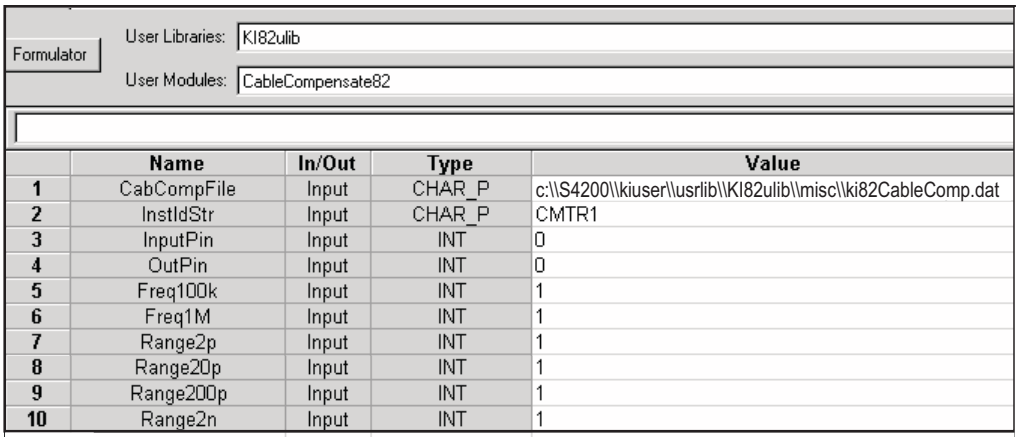
Figure E-15
Display-cap-file spread sheet showing capacitor source values

	A	B	C	D
1	DisplayCable	Range	Values100k	Values1M
2	0	2.00000E-12	4.77700E-13	4.77700E-13
3		2.00000E-12	1.46100E-12	1.46100E-12
4		2.00000E-11	4.79600E-12	4.79600E-12
5		2.00000E-11	1.78300E-11	1.78300E-11
6		2.00000E-10	4.66800E-11	4.66800E-11
7		2.00000E-10	1.81100E-10	1.81100E-10
8		2.00000E-09	4.75500E-10	4.75900E-10
9		2.00000E-09	1.77100E-09	1.77600E-09
10				

C. Perform cable compensation (CableCompensate82)

1. In the Project Navigator (Figure E-12), double-click **cable-compensate** or **cablecomp** to open the UTM. Figure E-16 shows the default parameters for the **CableCompensate82** user module.
2. Ensure that line 1 of the parameter list has the same file directory path that is used in step 1 (Figure E-13).
3. Enable / disable cable compensation: Lines 5 through 10 of the parameter list are used to either disable (0) or enable (1) cable compensation for the test frequencies and ranges. Figure E-16 shows cable compensation enabled for all ranges and test frequencies. Refer to the library reference section for the meaning of the input parameters.

Figure E-16
CableCompensate82 user module

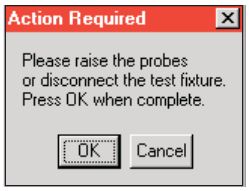


	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	Freq100k	Input	INT	1
6	Freq1M	Input	INT	1
7	Range2p	Input	INT	1
8	Range20p	Input	INT	1
9	Range200p	Input	INT	1
10	Range2n	Input	INT	1

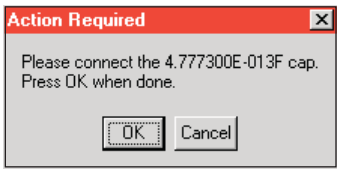
4. Execute the **cable-compensate** or **cablecomp** UTM by clicking the green **Run** button. A series of dialog boxes will guide you through the cable compensation process. The three basic dialog boxes are shown in Figure E-17:
   - **Figure E-17A:** This dialog box will appear when an offset (open circuit) measurement is required. Open the circuit as close to the DUT as possible.
   - **Figure E-17B:** This dialog box will instruct you to connect a capacitance source in place of the DUT. Note that the value in the dialog box corresponds to a calibration value entered by the user in step 1. Connect the capacitance source as close to the DUT as possible.
   - **Figure E-17C:** This dialog box compares the measured value to the calibration (nominal) value entered by the user. The two readings should be fairly close. If they are not, you probably connected the wrong capacitance source or had an open circuit condition. In that case, click **Cancel** to abort the cable compensation process.

NOTE: *Clicking **Cancel** in a cable compensation dialog box aborts the cable compensation process. You can start over by clicking the green **Run** button.*

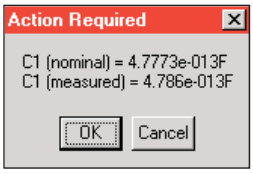
Figure E-17
Cable compensation dialog boxes



A. Measure offset



B. Measure capacitance source



C. Compare readings

Capacitance tests

QTsweep (equilibrium test)

In order to achieve accurate simultaneous C-V test results, measurements must be performed with the device in equilibrium. A device is considered to be in equilibrium when all internal capacitances are fully charged before measuring the capacitance. For a fully charged capacitor, any measured current is leakage.

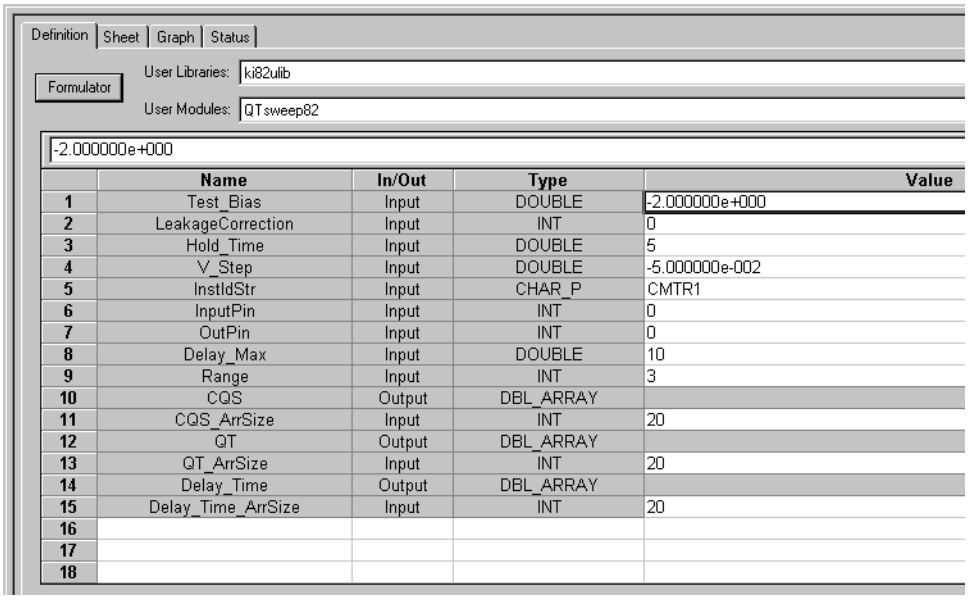
After voltage step is applied to the device, a delay time is used to allow capacitances to fully charge before measuring quasistatic capacitance. If the delay time is too short (capacitors still charging), the quasistatic capacitance measurement will not be accurate. This test allows you to determine the delay time required to achieve equilibrium.

This example assumes that the Model 82 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a substrate on a wafer. Perform the following steps to perform the equilibrium test:

Open and execute QTsweep UTM

1. In the Project Navigator for the **simcv** project plan (Figure E-12), double-click **QTsweep** to open the UTM (Figure E-18). From the **Definition** tab, you can modify the test parameters, if desired. Refer to the library reference section for the meaning of the input parameters. If you use the parameters shown in Figure E-18, the Model 82 will perform 20 quasistatic capacitance measurements using 20mV pulses (V_Step) ranging from 0.07 seconds (the default minimum) to 1 second (Delay_Max) in time duration.
2. Execute the test by clicking the green **Run** button.

Figure E-18
QtSweep82 user module (equilibrium test)



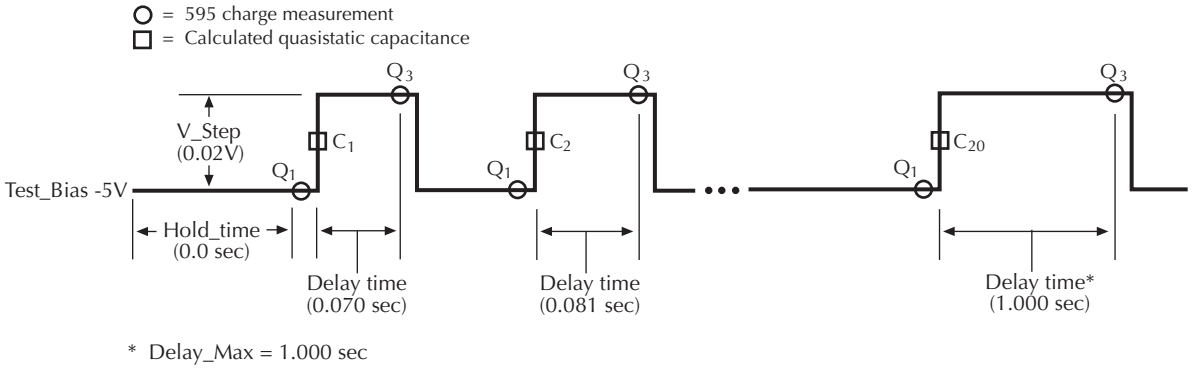
	Name	In/Out	Type	Value
1	Test_Bias	Input	DOUBLE	-2.000000e+000
2	LeakageCorrection	Input	INT	0
3	Hold_Time	Input	DOUBLE	5
4	V_Step	Input	DOUBLE	-5.000000e-002
5	InstIdStr	Input	CHAR_P	CMTR1
6	InputPin	Input	INT	0
7	OutPin	Input	INT	0
8	Delay_Max	Input	DOUBLE	10
9	Range	Input	INT	3
10	CQS	Output	DBL_ARRAY	
11	CQS_ArrSize	Input	INT	20
12	QT	Output	DBL_ARRAY	
13	QT_ArrSize	Input	INT	20
14	Delay_Time	Output	DBL_ARRAY	
15	Delay_Time_ArrSize	Input	INT	20
16				
17				
18				

Equilibrium test (QTsweep) description

This test performs a quasistatic capacitance measurement (C_{QS}) using 20 different delay times. The voltage bias and pulse amplitude are held constant during the test. The current (Q / t) at the end of each reading sample is also calculated ($i = \Delta Q/\Delta t$).

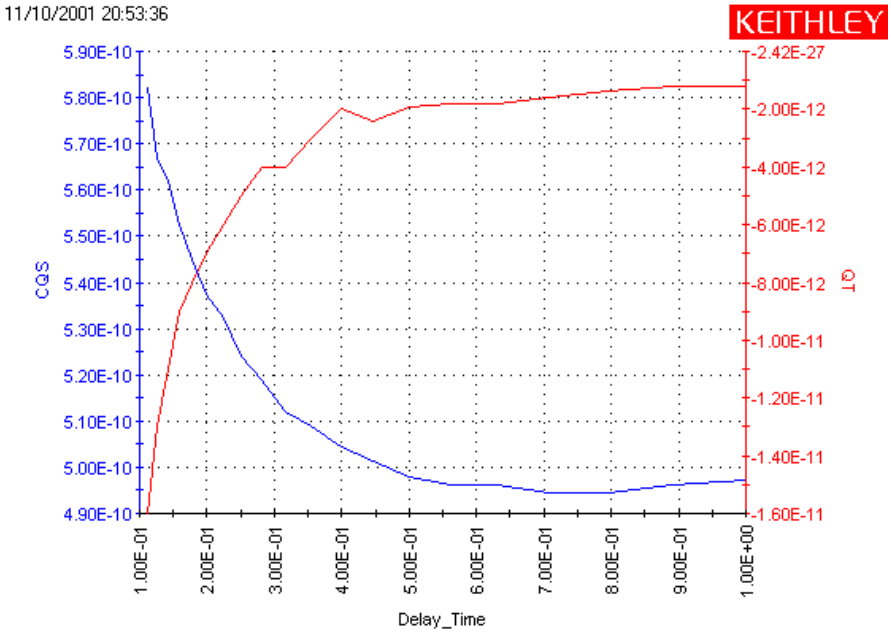
Figure E-19 shows the pulse stream for the equilibrium test using the parameters shown in Figure E-19. As shown, the last reading sample uses a set delay (Delay_Max) of one second, while the first reading sample uses a delay of 70 msec (which is the minimum). The delay times for the other 18 reading samples are then automatically set. After the first pulse, each subsequent pulse delay time increases logarithmically in progression up to the maximum delay (Delay_Max).

Figure E-19
Equilibrium test



The generated graph for this test plots quasistatic capacitance (C_{QS}) vs. delay time, and leakage current (Q / t) vs. delay time. A typical graph for the equilibrium test is shown in Figure E-20. The optimal delay time occurs when both curves flatten out to a slope of zero. For maximum accuracy, choose the second point on the curves after they have flattened out.

Figure E-20
Equilibrium test graph



Simultaneous C-V sweep

The Model 82 uses the Models 595 and 590 to perform simultaneous C-V measurements. Details on simultaneous C-V measurements are provided in Key concepts, Simultaneous C-V measurements.

This example assumes that the Model 82 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a substrate on a wafer. Perform the following steps to perform a simultaneous C-V sweep.

Open and execute cvsweep UTM

1. In the Project Navigator (Figure E-12), double-click **cvsweep** to open the UTM (Figure E-21). From the **Definition** tab, you can modify the test parameters, if desired. Refer to the library reference section for the meaning of the input parameters. If you use the parameters as shown in Figure E-21, the Model 82 will perform a -3 to +3V staircase sweep using 20 mV steps, delaying 70 msec on each step.
2. Execute the test by clicking the green run button.

Figure E-21
SIMCVsweep82 user module

Definition | Sheet | Graph | Status

Formulator

User Libraries: ki82ulib

User Modules: SIMCVsweep82

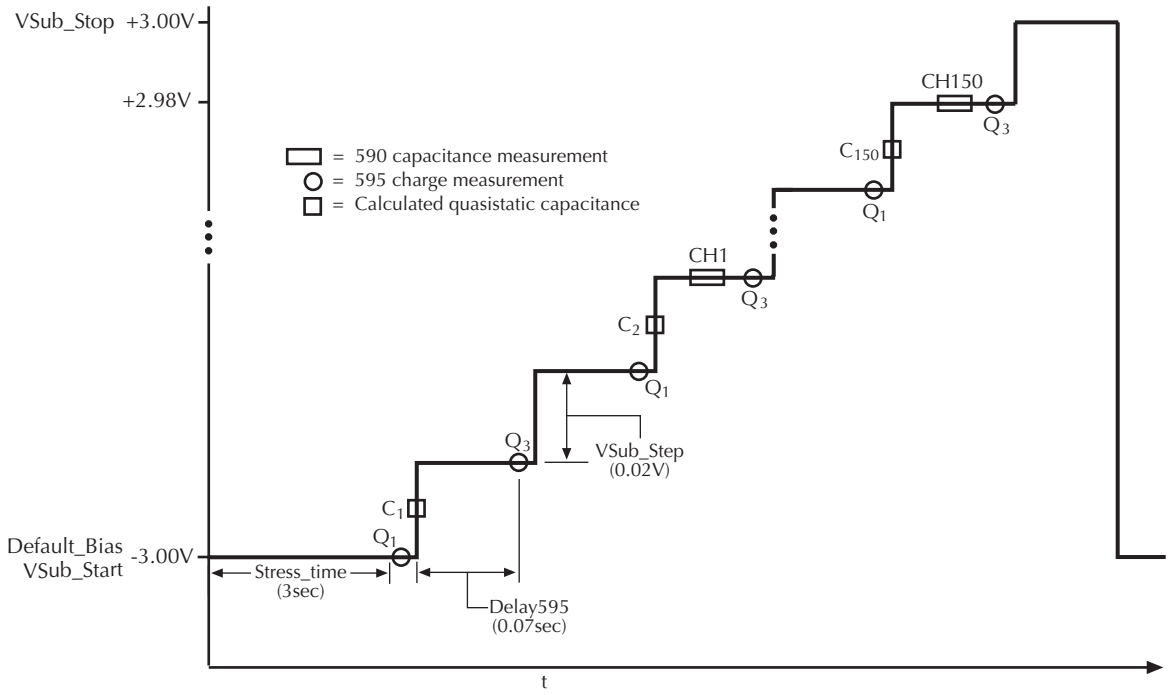
0

	Name	In/Out	Type	Value
1	Frequency	Input	INT	0
2	Default_Bias	Input	DOUBLE	-3.000000e+000
3	Stress_Time	Input	DOUBLE	0.000000e+000
4	VSub_Start	Input	DOUBLE	-2.000000e+000
5	VSub_Stop	Input	DOUBLE	4.000000e+000
6	VSub_Step	Input	DOUBLE	5.000000e-002
7	Range595	Input	INT	2
8	Range590	Input	INT	4
9	Model590	Input	INT	0
10	Filter	Input	INT	0
11	Delay595	Input	DOUBLE	5.000000e-001
12	LeakageCorrection	Input	INT	1
13	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
14	OffsetCorrect	Input	INT	0
15	InstIdStr	Input	CHAR_P	CMTR1
16	InputPin	Input	INT	0
17	OutPin	Input	INT	0
18	CHF	Output	DBL_ARRAY	
19	CHF_ArrSize	Input	INT	500
20	VSub	Output	DBL_ARRAY	
21	VSub_ArrSize	Input	INT	500
22	CQS	Output	DBL_ARRAY	
23	CQS_ArrSize	Input	INT	500
24	G_or_R	Output	DBL_ARRAY	
25	G_or_R_ArrSize	Input	INT	500
26	QT	Output	DBL_ARRAY	
27	QT_ArrSize	Input	INT	500

cvsweep test description

As shown in Figure E-21, the **cvsweep** UTM uses the **SIMCVsweep82** user module to perform simultaneous C-V measurements. As previously explained, a Model 595 quasistatic measurement is a two-step process requiring at least two charge measurements. As shown in Figure E-22, charge measurements on two steps are performed to yield a single quasistatic reading. The Model 590 performs a capacitance measurement on every second step of the staircase sweep.

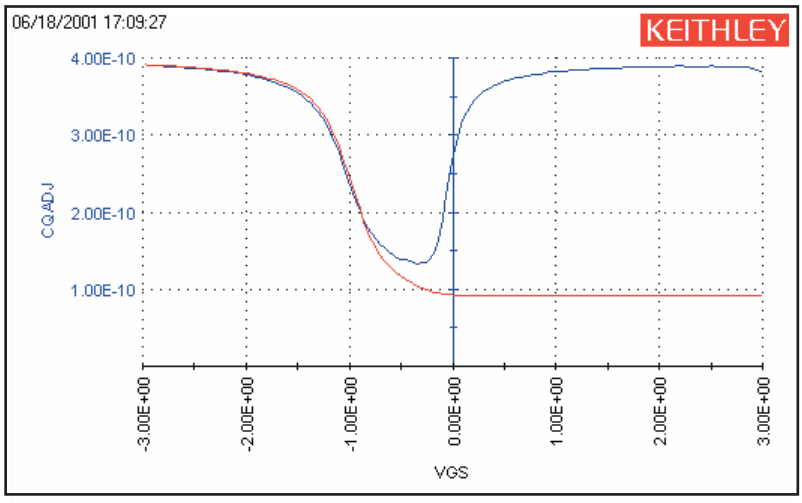
Figure E-22
Simultaneous C-V linear staircase sweep



The graph for this test plots Model 590 capacitance (red trace) and Model 595 quasistatic capacitance (blue trace) vs. bias voltage. Figure E-23 shows a typical graph that is generated by this test.

NOTE: *The shape of the curves in Figure E-23 indicate that measurements were performed with the device in equilibrium. If the curves for your test deviate significantly, then the device was probably not in equilibrium. Perform the equilibrium test (**QTsweep**) to determine the optimum delay time (**Delay595** parameter) to use for the simcv test (**SIMCVsweep82** user module).*

Figure E-23
cvsweep graph



C-t sweep

The Model 82 uses the Model 590 to perform a specified number of capacitance measurements using a specified time interval between reading samples. The specified voltage bias is held constant for this test. Details on simultaneous C-t measurements are provided in Key concepts, C-t measurements.

This example assumes that the Model 82 is connected directly to the DUT. The DUT could be a device installed in a test fixture, or a substrate on a wafer. Perform the following steps to perform a CtSweep.

Open and execute the CtSweep UTM

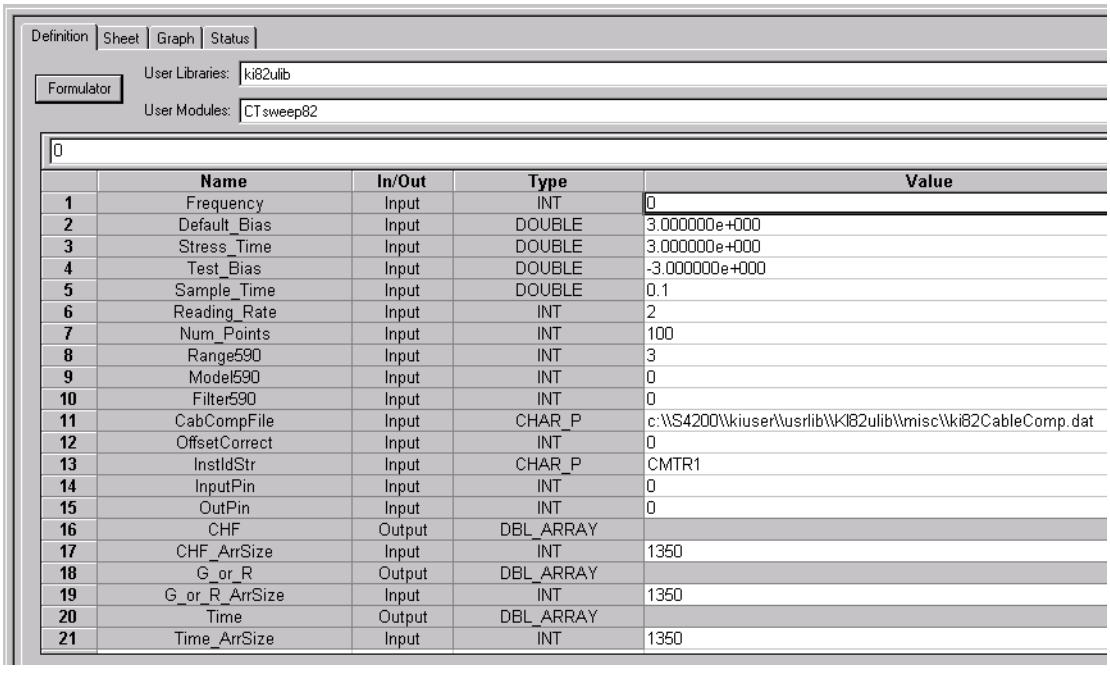
1. In the Project Navigator (Figure E-12), double-click **CtSweep** to open the UTM (Figure E-24). From the **Definition** tab, you can modify the test parameters, if desired. If you use the parameters shown in Figure E-24, the Model 82 will perform 100 capacitance measurements.
2. Execute the test by clicking the green **Run** button.

CtSweep test description

As shown in Figure E-24, the **CtSweep** UTM uses the **CTsweep82** user module to perform C-t measurements. Refer to the library reference section for the meaning of the input parameters. If using the parameters shown in Figure E-24, the Model 590 performs 100 capacitance measurements using a 100 msec sample time between reading samples. The time interval between reading samples is determined by the set sample time and the selected reading rate.

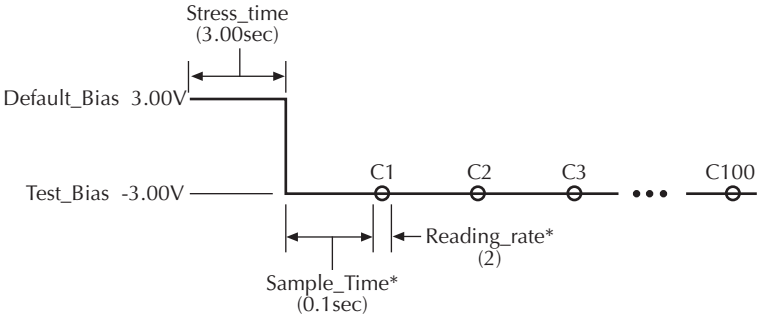
The graph for this test plots capacitance vs. time as shown in Figure E-26.

Figure E-24
CtSweep82 user module



	Name	In/Out	Type	Value
1	Frequency	Input	INT	0
2	Default_Bias	Input	DOUBLE	3.000000e+000
3	Stress_Time	Input	DOUBLE	3.000000e+000
4	Test_Bias	Input	DOUBLE	-3.000000e+000
5	Sample_Time	Input	DOUBLE	0.1
6	Reading_Rate	Input	INT	2
7	Num_Points	Input	INT	100
8	Range590	Input	INT	3
9	Model590	Input	INT	0
10	Filter590	Input	INT	0
11	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
12	OffsetCorrect	Input	INT	0
13	InstIdStr	Input	CHAR_P	CMTR1
14	InputPin	Input	INT	0
15	OutPin	Input	INT	0
16	CHF	Output	DBL_ARRAY	
17	CHF_ArrSize	Input	INT	1350
18	G_or_R	Output	DBL_ARRAY	
19	G_or_R_ArrSize	Input	INT	1350
20	Time	Output	DBL_ARRAY	
21	Time_ArrSize	Input	INT	1350

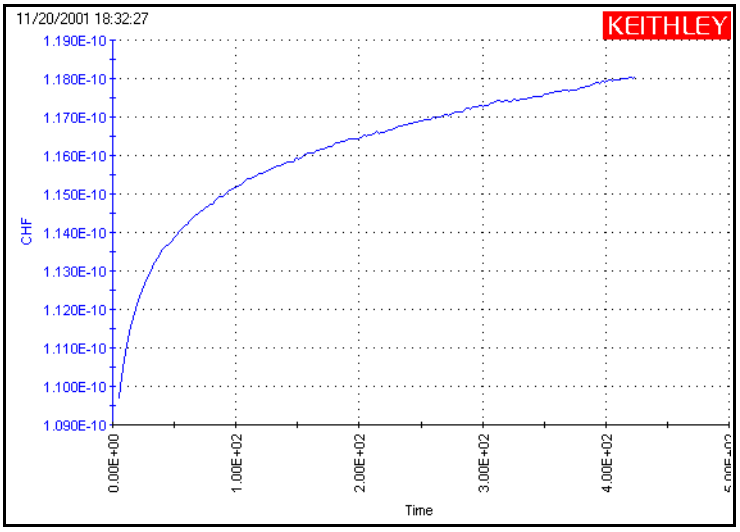
Figure E-25
C-t measurements



* Time interval between reading samples is the sum of set Sample Time and the Reading Rate time. The reading rate time for Reading_rate 2 is 1/18 sec (or 0.0555 sec). Therefore:

Time Interval = Sample Time + Reading Rate time
= 0.100 + 0.056
= 0.156 seconds

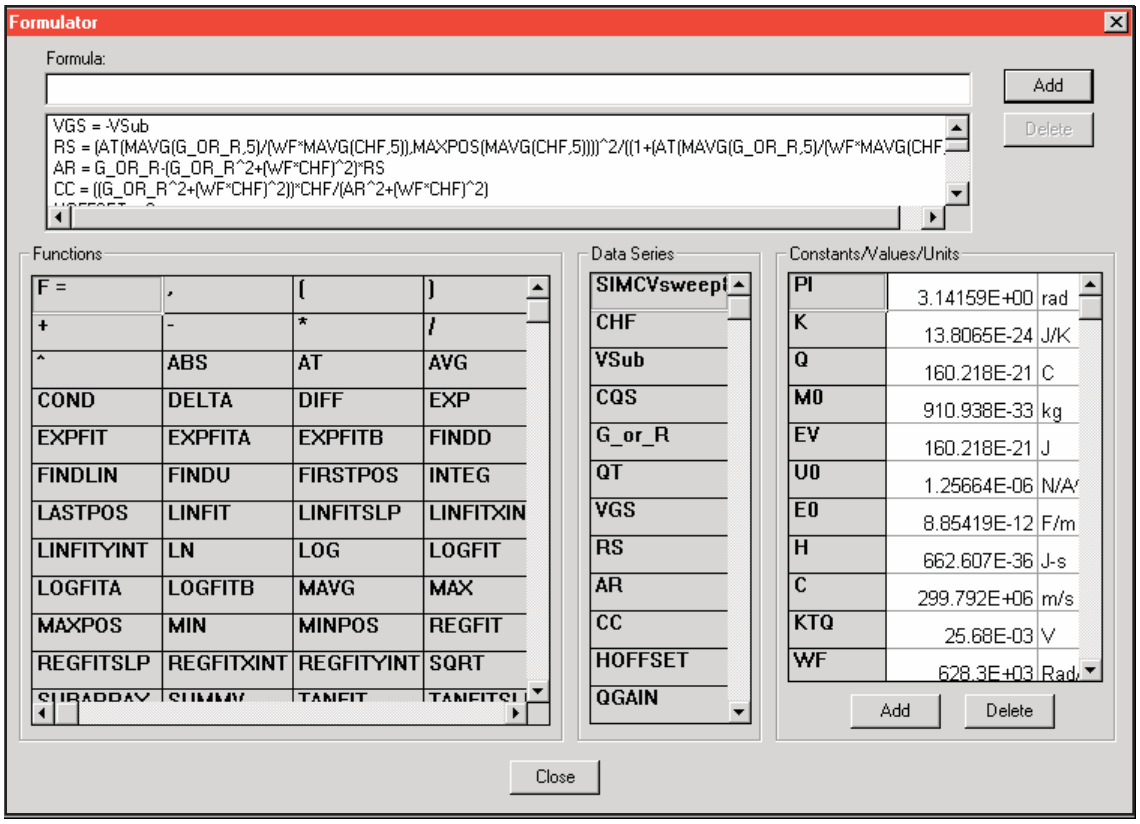
Figure E-26
CtSweep graph



Formulas for capacitance tests

Formulas to calculate data for graphs are located in the Formulator for each test. A Formulator window is opened by clicking the **Formulator** button on the Definition tab for the selected test. Figure E-27 shows the Formulator for the cvsweep test used in the **simcv** project.

Figure E-27
Formulator for cvsweep test (simcv project)



Formulas for the **cvsweep** test (**simcv** project), CtSweep test (lifetime project), and **cvsweep** test (**STVS** project) are shown in Table E-2, Table E-3, and Table E-4, respectively.

NOTE: *Refer to "Simultaneous C-V analysis" for details on simultaneous C-V theory and the formulas.*

NOTE: *The values for constants used in the formulas are located in the Constants / Values / Units area in the Formulator. Examples:*

ConstantValueUnits

AREA0.012mm^2

EOX (ε_{OX})3.4 x 10^{-13}F/cm

ES (ε_S)1.04 x 10^{-12}F/cm

Table E-2
Formulas for cvsweep test (simcv project)

Formula name	Description and formula
VGS	Gate voltage: VGS = -VSub
RS	Serial resistance calculated by high frequency CV: RS = (AT(MAVG(G_OR_R,5)/(WF*MAVG(CHF,5)),MAXPOS(MAVG(CHF,5))))^2/ ((1+(AT(MAVG(G_OR_R,5)/(WF*MAVG(CHF,5)),MAXPOS(MAVG(CHF,5))))^2) *(AT(MAVG(G_OR_R,5),MAXPOS(MAVG(CHF,5))))
AR	Intermediate parameter for calculation of CC: AR = G_OR_R-(G_OR_R^2+(WF*CHF)^2)*RS
CC	Corrected high frequency capacitance by compensating serial resistance: CC = ((G_OR_R^2+(WF*CHF)^2))*CHF/(AR^2+(WF*CHF)^2)
HOFFSET	Offset for high frequency capacitance (entered by user): HOFFSET = 0
QGAIN	Gain for quasistatic capacitance (entered by user): QGAIN = 1
QOFFSET	Offset for quasistatic capacitance (entered by user): QOFFSET = 0
CQADJ	Adjusted quasistatic capacitance by using QGAIN and QOFFSET: CQADJ = QGAIN*CQS+QOFFSET
HGAIN	Gain for calculated high frequency capacitance that is calculated: HGAIN = AT(MAVG(CQS,5)/MAVG(CC,5),MAXPOS(MAVG(CC,5)))
CHADJ	Adjusted high frequency capacitance by using HGAIN and HOFFSET: CHADJ = HGAIN*CC+HOFFSET
COX	Oxide capacitance: COX = MAX(MAVG(CHADJ,5))+1E-15
CMIN	Minimum capacitance from high frequency: CMIN = MIN(MAVG(CHADJ,5))+1E-15
TOXNM	Calculated thickness of oxide (in nanometers): TOXNM = 1E7*AREA*EOX/COX
INVCSQR	Inversed square of high frequency capacitance: INVCSQR = 1/(MAVG(CHADJ,5))^2
STRETCHOUT	Stretch out factor due to interfacial states: STRETCHOUT = MAVG((1-CQADJ/COX)/(1-CHADJ/COX),5)
NDOPING	Doping density: NDOPING = ABS(-2*STRETCHOUT/(AREA^2*Q*ES)/(DELTA(INVCSQR)/DELTA(VGS)))
DEPTHM	Depletion depth (in meters): DEPTHM = 1E-2*AREA*ES*(1/CHADJ-1/COX)
N90W	Doping density at 90% of maximum depletion depth: N90W = AT(NDOPING,FINDLIN(DEPTHM,0.9*MAX(DEPTHM),2))
DEBYEM	Debye length (in meters): DEBYEM = SQRT(ES*K*TEMP/(ABS(N90W)*Q^2))*1E-2
CFB	Flatband capacitance: CFB = (COX*ES*AREA/(DEBYEM*1E2))/(COX+(ES*AREA/(DEBYEM*1E2)))
VFB	Flatband voltage: VFB = AT(VGS,FINDLIN(CHADJ,CFB,2))
PHIB	Bulk potential:

Table E-2 (continued)
Formulas for cvsweep test (simcv project)

Formula name	Description and formula
	PHIB = (-1)*K*TEMP/Q*LN(ABS(N90W)/NI)*DOPETYPE
VTH	Threshold voltage: VTH = VFB+DOPETYPE*(AREA/COX*SQRT(4*ES*Q*ABS(N90W*PHIB))+ 2*ABS(PHIB))
WMS	Work function difference between metal and semiconductor: WMS = WM-(WS+(EBG/2)-PHIB)
QEFF	Effective charge in oxide: QEFF = COX*(WMS-VFB)/AREA
BEST_LO	Index from DEPTHM array that is three Debye lengths from the surface: BEST_LO = FINDD(DEPTHM,3*DEBYEM,2)
BEST_HI	Index from DEPTHM array that is 95% of maximum depletion length, or twice the screening length in the semiconductor, whichever is larger: BEST_HI = FINDD(DEPTHM,COND(2*DEBYEM*SQRT(LN(ABS(N90W/NI))), MAX(DEPTHM),2*DEBYEM*SQRT(LN(ABS(N90W/NI))),0.95*MAX(DEPTHM)), 2)
NAVG	Average doping calculated between index BEST_HI and BEST_LO: NAVG = AVG(SUBARRAY(NDOPING,COND(BEST_HI, BEST_LO, BEST_HI, BEST_LO),COND(BEST_HI, BEST_LO,BEST_LO,BEST_HI)))
DIT	Interfacial states density: DIT = 1/(AREA*Q)*(1/(1/CQADJ-1/COX)-1/(1/CHADJ-1/COX))
PSISPSIO	PSIS - PSIO, which is surface potential: PSISPSIO = SUMMV((1-CQADJ/COX)*DELTA(VGS))*DOPETYPE
PSIO	Offset in surface potential due to calculation method and flatband voltage: PSIO = AT(PSISPSIO,FINDLIN(VGS,VFB,2))
PSIS	Silicon surface potential. More precisely, this value represents band bending and is related to surface potential via the bulk potential: PSIS = PSISPSIO-PSIO
EIT	Interface trap energy with respect to mid band gap: EIT = PSIS+PHIB

Table E-3
Formulas for CtSweep test (lifetime project)

Formula name	Description and formula
NAVG	Average doping: NAVG = 1E15
COX	Oxide capacitance (in picofarads): COX = 450
WF	Equilibrium inversion depth (in centimeters): WF = ES*AREA*(1/MAX(CHF)-1E12/COX)
WWF	W - WF, where W is the depletion depth (in centimeters): WWF = ES*AREA*(1/CHF-1E12/COX)-WF
GNI	Generation rate in S^{-1} divided by intrinsic carrier concentration: GNI = -(ES*AREA*NAVG*COX/1E12)*DIFF(1/CHF^2,TIME)/NI

Table E-4
Formulas for cvsweep test (STVS project)

Formula name	Description and formula
VGS	Gate voltage: VGS = -VSub
RS	Serial resistance calculated by high frequency CV: RS = AT(MAVG(G_OR_R,5)/(WF*MAVG(CHF,5)),MAXPOS(MAVG(CHF,5))))^2/ ((1+(AT(MAVG(G_OR_R,5)/(WF*MAVG(CHF,5)),MAXPOS(MAVG(CHF,5))))^2) *(AT(MAVG(G_OR_R,5),MAXPOS(MAVG(CHF,5)))))
AR	Intermediate parameter for calculation of CC: AR = G_OR_R-(G_OR_R^2+(WF*CHF)^2)*RS
CC	Corrected high frequency capacitance by compensating serial resistance: CC = ((G_OR_R^2+(WF*CHF)^2))*CHF/(AR^2+(WF*CHF)^2)
HOFFSET	Offset for high frequency capacitance (entered by user): HOFFSET = 0
DELAY	595 delay time: DELAY = 0.15
HGAIN	Gain for calculated high frequency capacitance that is calculated: HGAIN = AT(MAVG(CQS,5)/MAVG(CC,5),MAXPOS(MAVG(CC,5)))
CHADJ	Adjusted high frequency capacitance by using HGAIN and HOFFSET: CHADJ = HGAIN*CC+ HOFFSET
VSTEP	595 step voltage: VSTEP = 0.02
LEAKSLP	Average slop of leakage current neglecting the contribution of mobile ion: LEAKSLP = LINEFITSLP(VGS, QT, 49, 200) 49 and 200 are indexes on QT array to fit the slope.
QGAIN	Gain for quasistatic capacitance (entered by user): QGAIN = 1
CQADJ	Adjusted quasistatic capacitance by using QGAIN and QOFFSET: CQADJ = QGAIN*CQS+QOFFSET
NM	Mobile ion density: NM = AVG((CQADJ- CHADJ)*ABS(DELTA(VGS))*(LASTPOS(DELTA(VGS))- FIRSTPOS(DELTA(VGS)))/Q/AREA

Choosing the right parameters

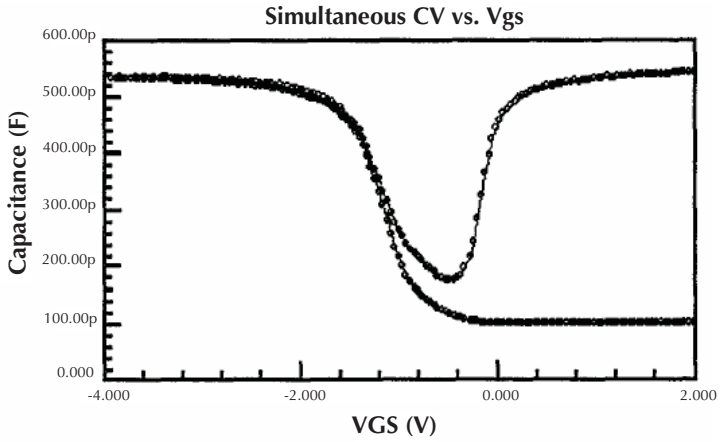
Optimal C-V measurement parameters

Simultaneous C-V measurement is a complicated matter. Besides system considerations, you should carefully choose the measurement parameters. Refer to the following discussion for considerations when selecting these parameters.

Start, stop, and step voltages

Most C-V data is derived from the sweep transition, or depletion region of the C-V curve. For that reason, start and stop voltages should be chosen so that the depletion region makes up about 1/3 to 2/3 of the voltage range (see Figure E-28).

Figure E-28
Typical simultaneous C-V curve



The upper flat, or accumulation region of the high frequency C-V curve defines the oxide capacitance, C_{OX}. Since most analysis relies on the ratio C/C_{OX}, it is important that you choose a start or stop voltage (depending on the sweep direction) to bias the device into strong accumulation at the start or the end of the sweep.

You should carefully consider the size of the step voltage. Start, stop, and step size determine the total number of data points in the sweep. Some compromise is necessary between having too few data points in one situation, or too many data points in the other.

For example, the complete doping profile is derived from data taken in the depletion region of the curve by using a derivative calculation. As the data point spacing decreases, the vertical point spacing is increasingly caused by noise rather than changes in the desired signal. Consequently, choosing too many points in the sweep will result in increased noise rather than an increased resolution in C-V measurement. It also takes more time to perform a C-V sweep.

Many calculations depend on good measurements in the depletion region, and too few data points in this region will give poor results. A good compromise results from choosing parameters that will yield a capacitance change per step of approximately ten times the error in the signal.

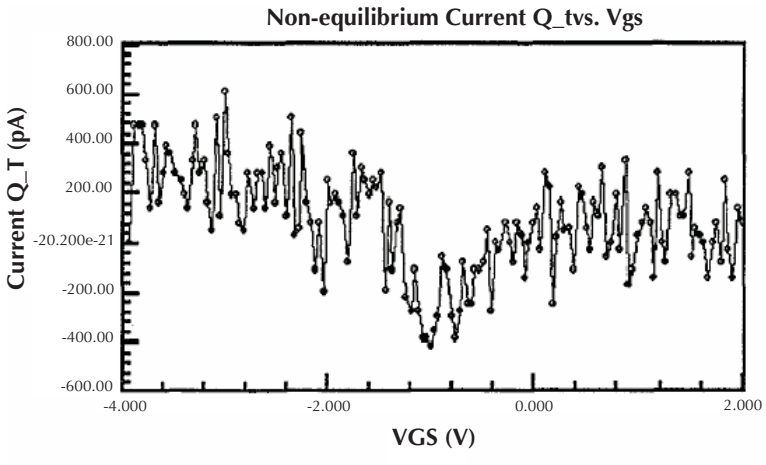
Sweep Direction

For C-V sweeps, you can sweep either from accumulation to inversion, or from inversion to accumulation. Sweeping from accumulation to inversion will allow you to achieve deep depletion, profiling deeper into the semiconductor than you otherwise would obtain by maintaining equilibrium. When sweeping from inversion to accumulation, you should use a light pulse to achieve equilibrium more rapidly before the sweep begins.

Delay Time

For accurate measurement, delay time must be carefully chosen to ensure that the device remains in equilibrium in the inversion region during a sweep. With too fast a sweep, the device will remain in non-equilibrium, affecting Q/t (Figure E-29), and also resulting in skewed C-V curves.

Figure E-29
Leakage current Q/t through device



Determining the optimal delay time

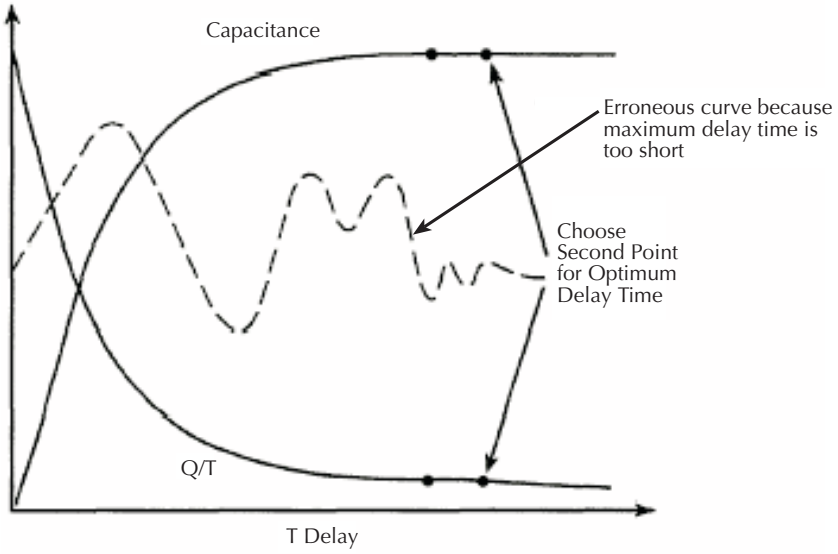
For accurate interface trap density measurement, delay time must be carefully chosen to ensure that the device remains in equilibrium in the inversion region during a sweep. An equilibrium test is provided to determine the optimum delay time.

The equilibrium test uses the Model 595 to perform a series of quasistatic capacitance and Q/t current measurements using different delay times. Figure E-30 shows the typical capacitance and Q/t curves generated for this test. As shown, the optimal delay is the second TDelay point after both curves have flattened out.

For long delay times, the measurement process can become very long with some devices. You may be tempted to speed up the test by using a shorter delay time. However, doing so is not recommended since it is difficult to quantify the amount of accuracy degradation in any given situation.

NOTE: *See "Model 82 project plans, QTsweep (equilibrium test)" for details on performing the equilibrium test.*

Figure E-30
Choosing optimal delay time



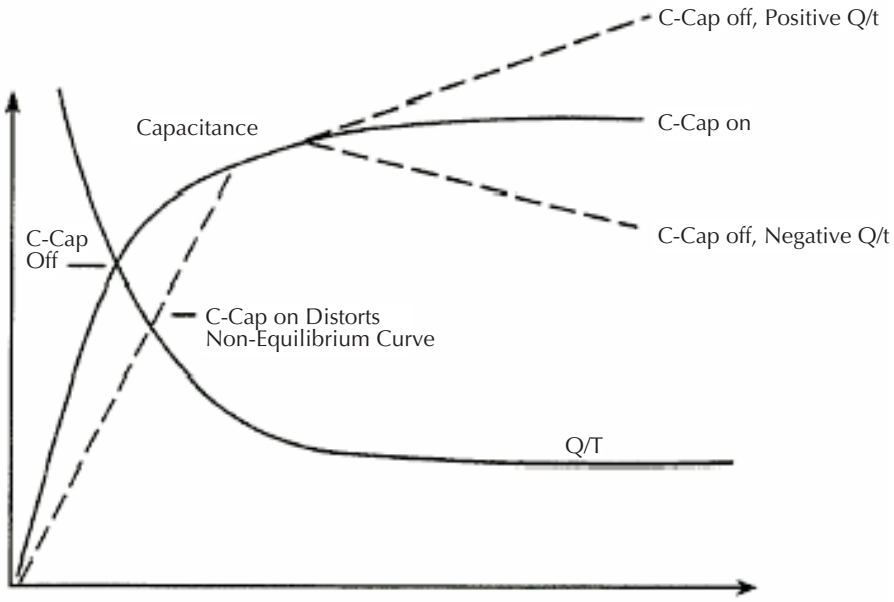
Determining delay time with leaky devices

When testing for delay time on devices with relatively large leakage currents, it is recommended that you use the corrected capacitance feature, which is designed to compensate for leakage current. The reason for doing so is illustrated in Figure E-31. When large leakage currents are present, the capacitance curve will not flatten out in equilibrium, but will instead either continue to rise (positive Q/t) or begin to decay (negative Q/t).

Using corrected capacitance results in the normal flat capacitance curve in equilibrium due to leakage compensation. Note, however, that the curve taken with corrected capacitance will be distorted in the non-equilibrium region, so data in that region should be considered to be invalid when using corrected capacitance. If it is necessary to use corrected capacitance when determining delay time, it is recommended that you make all measurements on that particular device using corrected capacitance.

NOTE: *Corrected capacitance can be enabled for simultaneous C-V measurements by setting the "LeakageCorrection" parameter to "1" (see line 12 of the SIMCVsweep82 user module).*

Figure E-31
Capacitance and leakage current curves of leaky device



Testing slow devices

A decaying noise curve, such as the dotted line shown in Figure E-30, will result if the maximum delay time is too short for the device being tested. This phenomenon, which is most prevalent with slow devices, occurs because the signal range is too small. To eliminate such erroneous curves, choose a longer maximum delay time. A good starting point for unknown devices is a 30-second maximum delay time.

Correcting residual errors

Controlling errors at the source is the best way to optimize C-V measurements, but doing so is not always possible. Remaining residual errors include offset, gain, noise, and voltage-dependent errors. Ways to deal with these error sources are discussed in the following paragraphs.

Offsets

Offset capacitance and conductance caused by the test apparatus can be eliminated by performing a suppression with the probes in the up position. These offsets will then be nulled out when the measurement is made. Whenever the system configuration is changed, the suppression procedure should be repeated. For maximum accuracy, it is recommended that you perform a probes-up suppression or at least verify prior to every measurement.

NOTE: *Suppression can be enabled for simultaneous C-V measurements by setting the "OffsetCorrect" parameter to "1" (see line 14 of the SIMCVsweep82 user module).*

Gain and nonlinearity errors

Gain errors are difficult to quantify. For that reason, gain correction is applied to every measurement. Gain constants are determined by measuring accurate calibration sources during the cable correction process.

Nonlinearity is normally more difficult to correct for than are gain or offset errors. The cable correction provides nonlinearity compensation for high-frequency measurements, even for non-ideal configurations such as switching matrices.

Voltage-dependent offset

Voltage-dependent offset (curve tilt) is the most difficult to correct error associated with quasistatic C-V measurements. It can be eliminated by enabling corrected capacitance (LeakageCorrection parameter set to 1). In this technique, the current flowing in the device is measured as the capacitance value is measured. The current is known as Q/t because its value is derived from the slope of the charge integrator waveform. Q/t is used to correct capacitance readings for offsets caused by shunt resistance and leakage currents.

Care must be taken when using the corrected capacitance feature, however. When the device is in non-equilibrium, device current adds to any leakage current, with the result that the curve is distorted in the non-equilibrium region. The solution is to keep the device in equilibrium throughout the sweep by carefully choosing the delay time.

Noise

Residual noise on the C-V curve can be minimized by using filtering when taking your data. The Filter parameter sets the filter (see line 10 of SIMCVsweep82 user module). However, the filter will reduce the sharpness of the curvature in the transition region of the quasistatic curve depending on the number of data points in the region. This change in the curve can cause C_Q to dip below C_H resulting in erroneous D_{IT} calculations. If this situation occurs, turn off the filter or add more data points.

ki82ulib user library reference

The user modules in the ki82ulib user library are used to control the Model 82 C-V System. These user modules are summarized in Table E-5. Also listed in the table are the Keithley Instruments-created UTM names that use the user modules.

Details for each of the user modules follow the table.

Table E-5
Model 82 C-V System user modules

User module	UTM name(s)	Description
CableCompensate82	cable-compensate cablecomp	Performs cable compensation using known capacitance source values.
CTsweep82	CtSweep	Performs C-t measurements.
DisplayCableCompCaps82	display-cap-file	Places capacitance source values in a spread sheet.
QTsweep82	QtSweep	Performs quasistatic measurement sweep.
SaveCableCompCaps82	save-cap-file savecablecompfile	Saves entered capacitance source values in a file.
SIMCVsweep82	cvsweep	Performs simultaneous C-V sweep.

CableCompensate82 user module

Overview

This user module (see Figure E-32) is used to perform cable compensation for the selected ranges and test frequencies of the Model 590. For the input parameters shown in Figure E-32, cable compensation for the Model 590 will be performed for the 2pF, 20 pF, 200 pF, and 2 nF ranges. It will be performed for both the 100 kHz and 1 MHz test frequencies. The line 1 input parameter indicates the directory path where the user-input capacitor source values are saved. These values are entered and saved using the **SaveCableCompCaps82** user module.

User-entered parameters and returned outputs for this user module are explained in the User module description.

NOTE: *For details on the procedure to perform cable compensation, see "Model 82 project plans, Cable compensation tests."*

Figure E-32
CableCompensate82 user module

Formulator
User Libraries: KI82ulib
User Modules: CableCompensate82

	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	InstIdStr	Input	CHAR_P	CMTR1
3	InputPin	Input	INT	0
4	OutPin	Input	INT	0
5	Freq100k	Input	INT	1
6	Freq1M	Input	INT	1
7	Range2p	Input	INT	1
8	Range20p	Input	INT	1
9	Range200p	Input	INT	1
10	Range2n	Input	INT	1

User module description

The **CableCompensate82** routine performs Model 590 cable compensation using the capacitor values stored in the specified cable compensation file. The resultant compensation values generated by the compensation process are stored in the same file.

SYNTAX:

status = CableCompensate82(char *CabCompFile, char *InstIdStr, int InputPin, int OutPin, int Freq100 k, int Freq1M, int Range2p, int Range20 p, int Range200 p, int range2n);

PROCEDURE:

For each range and test frequency specified by the input parameters, the following will occur:

1. You will be prompted to open the circuit so that an offset capacitance measurement can be made.
2. Once the offset capacitance measurement is completed, you will be prompted to connect the low value capacitor for the selected range. The system will perform the low capacitor compensation.
3. Next, you will be prompted to connect the high value capacitor for the selected range. The system will perform the high value capacitor compensation.
4. You will then be prompted to reconnect the low capacitor.
5. The nominal and measured values will be displayed in a dialog box. If you are unsatisfied with the measurement, press cancel to abort the procedure. If you press cancel, the cable compensation file will not be affected.

When all selected ranges and frequencies have been compensated successfully, the cable compensation values will be saved.

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist, or there is no path specified (null string), the default compensation parameters will be used. When entering the path, be sure to use two '\' characters to separate each directory. For example, if your cable file is located in `C:\calfiles\82cal.dat`, you would enter the following:

```
C:\\calfiles\\82cal.dat
```

InstIdStr (char *) The CMTR instrument ID for the Model 82 system. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the Model 5951's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 5951 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72. See the following **NOTE**.

NOTE: *If a switch matrix to route signals is being controlled by a connection UTM (for example, **"connect"**), there is no need to connect InputPin and OutputPin. Set these parameters to 0.*

OutPin (int) The DUT pin to which the Model 5951's output terminal will be attached. If a value less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 5951 output will be connected to the specified pin. Valid values for this parameter are -1 to 72. See the previous **NOTE**.

Freq100 k (int) A flag indicating whether to perform the compensation procedure for the 100 kHz frequency. 0 indicates skip the compensation procedure for this frequency, while 1 indicates perform the compensation procedure for this frequency.

Freq1M	(int) A flag indicating whether to perform the compensation procedure for the 1 MHz frequency. 0 indicates skip the compensation procedure for this frequency, while 1 indicates perform the compensation procedure for this frequency.
Range2p	(int) A flag indicating whether to perform the compensation procedure for the 2pF range. 1 indicates that compensation should be performed for the 2pF range, while 0 indicates compensation should be skipped.
Range20 p	(int) A flag indicating whether to perform the compensation procedure for the 20 pF range. 1 indicates that compensation should be performed for the 20 pF range, while 0 indicates compensation should be skipped.
Range200 p	(int) A flag indicating whether to perform the compensation procedure for the 200 pF range. 1 indicates that compensation should be performed for the 200 pF range, while 0 indicates compensation should be skipped.
Range2n	(int) A flag indicating whether to perform the compensation procedure for the 2nF range. 1 indicates that compensation should be performed for the 2nF range, while 0 indicates compensation should be skipped.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0	OK.
-10000	(INVAL_INST_ID) The specified instrument ID does not exist.
-10021	(COMP_FILE_NOT_EXIST) The specified compensation file does not exist.
-10022	(KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.
-10090	(GPIB_ERROR_OCCURRED) There was a GPIB communications error.
-10100	(INVAL_PARAM) An invalid parameter was specified.

CtSweep82 user module

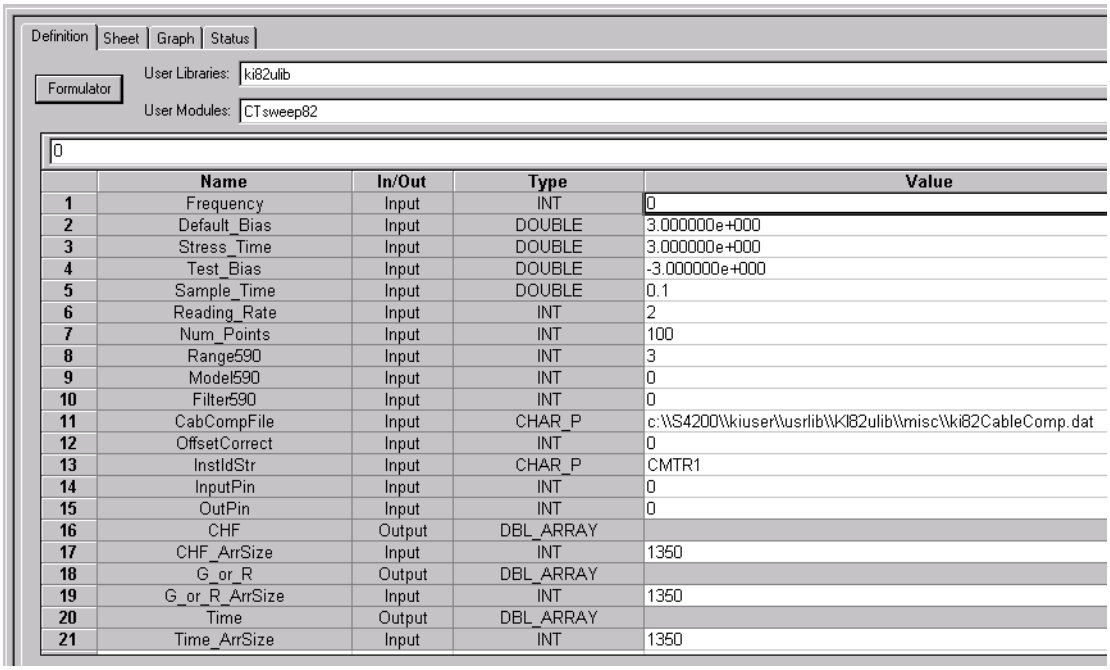
Overview

This user module performs a capacitance versus time sweep. Figure E-33 shows the default parameters for the **ctsweep** UTM which uses the **CtSweep82** user module. In this example, the Model 82 is set to first stress the DUT at +3V for three seconds, and then perform 100 capacitance measurements at -3V using a 0.1sec time interval (see Figure E-25).

User-entered parameters and outputs for this user module are explained in the User module description.

NOTE: *For details on C-t measurements, see "Model 82 project plans, C-t sweep."*

Figure E-33
CtSweep82 user module



	Name	In/Out	Type	Value
1	Frequency	Input	INT	0
2	Default_Bias	Input	DOUBLE	3.000000e+000
3	Stress_Time	Input	DOUBLE	3.000000e+000
4	Test_Bias	Input	DOUBLE	-3.000000e+000
5	Sample_Time	Input	DOUBLE	0.1
6	Reading_Rate	Input	INT	2
7	Num_Points	Input	INT	100
8	Range590	Input	INT	3
9	Model590	Input	INT	0
10	Filter590	Input	INT	0
11	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
12	OffsetCorrect	Input	INT	0
13	InstIdStr	Input	CHAR_P	CMTR1
14	InputPin	Input	INT	0
15	OutPin	Input	INT	0
16	CHF	Output	DBL_ARRAY	
17	CHF_ArrSize	Input	INT	1350
18	G_or_R	Output	DBL_ARRAY	
19	G_or_R_ArrSize	Input	INT	1350
20	Time	Output	DBL_ARRAY	
21	Time_ArrSize	Input	INT	1350

User module description

This module measures capacitance as a function of time at a certain bias. This method can be used for minority carrier lifetime measurements using Zerbst plot.

SYNTAX:

status = CtSweep82(int Frequency, double Default_Bias, double Stress_Time, double Test_Bias, double Sample_Time, int Reading_rate, int Num_Points, int Range590, int Model590, int Filter590, char *CabComFile, int OffsetCorrect, char *InstIdStr, int InputPin, int OutPin, double *CHF, int CHF_ArrSize, double *G_or_R, int G_or_R_ArrSize, double *Time, int Time_ArrSize);

PROCEDURE:

1. If desired, you will be prompted to open the circuit so that an offset capacitance measurement can be made.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.
3. A C-t sweep is performed.

INPUTS:

Frequency (int) A variable that selects the measurement frequency to use. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

Default_Bias (double) DC bias applied before and after a C-t sweep; -20 V to +20 (volts).

Stress_Time (double) Duration of the default bias before test bias is applied. Valid inputs are 0.001 to 65 seconds.

Test_Bias (double) Voltage bias for capacitance measurements; -20 V to +20 (volts).

Sample_Time (double) Time delay between each sampling measurement; 0.001 to 65 (seconds).

Reading_Rate (int) Selects the reading rate used to acquire the measurements. Valid inputs are 1 through 4 as shown below:

Table E-6
Reading rate valid inputs

Reading rate	Nominal reading rate	Readings	Display resolution
1	75 / sec	C,G,V	3.5 digits
2	18 / sec	C,G,V	4.5 digits
3	10 / sec	C,G,V	4.5 digits
4	1 / sec	C,G,V	4.5 digits

Num_Points (int) Number of sampling points; 1 to 1350.

Range590 (int) The measurement range for the Model 590 to use. The valid range of range values are 1 to 4:

Table E-7
Range590 valid range values

Range	100 kHz	1 MHz
1	2 pF / 2 uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2 mS
4	2 nF / 2 mS	2 nF / 20 mS

Model590 (int) Which measurement model to use for high frequency measurement. Entering 0 for this parameter will select the parallel model, while 1 will select the series model.

Filter590 (int) Enable / disable the analog filter. The analog filter can minimize the amount of noise that appears in the readings. It does, however, increase the measurement time. Entering 0 for this parameter will disable the filter; 1 will enable the filter.

CabCompFile (char *) The complete name and path of the cable compensation file. If this file does not exist, or this string is null, then cable compensation will not be used. The path that you specify must exist (when entering the path information, ensure that you use two '\\' characters to separate each directory level. For example, if your cable compensation file is located in file C:\calfiles\590cal.dat, you would enter C:\\calfiles\\590cal.dat).

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, then no offset measurement will be taken. If OffsetCorrect is 1, then an offset measurement will be made.

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the Model 5951's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the 5951 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72.

OutPin (int) The DUT pin to which the Model 5951's output terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made.

Otherwise, the 5951 output will be connected to the specified pin. Valid values for this parameter are -1 to 72.

CHF_ArrSize, (int) These values *must* be set equal to 1350.

G_or_R_ArrSize,

Time_ArrSize

OUTPUTS:

CHF (double *) The measured array of high frequency capacitance values.

G_or_R (double *) The array of measured conductance (G) or resistance (R) values.

Time (double) The array of Time from Model 595 output for each measurement step.

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0 OK.

-10000 (INVAL_INST_ID) The specified instrument ID does not exist.

-10020 (COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.

-10021 (COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-10045 (KI82_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

-10023 (KI590_MEAS_ERROR) A measurement error occurred.

-10090 (GPIB_ERROR_OCCURED) A GPIB communications error occurred.

-10091 (GPIB_TIMEOUT) A time-out occurred during communications.

-10100 (INVAL_PARAM) An invalid parameter was specified.

-10101 (ARRAY_SIZE_TOO_SMALL) The specified value for CHF_ArrSize, G_or_R_ArrSize, or Time_ArrSize was too small for the number of steps in the sweep.

-10102 (ERROR_PARSING) There was an error parsing the Model 590's response.

-10104 (USER_CANCEL) The user CANCELED the correction procedure.

DisplayCableCompCaps82 user module

Overview

This user module is used for Model 82 cable compensation. When this test is run, the nominal capacitance source values saved by the **SaveCableCompCaps82** user module, are placed into a spread sheet for convenient viewing.

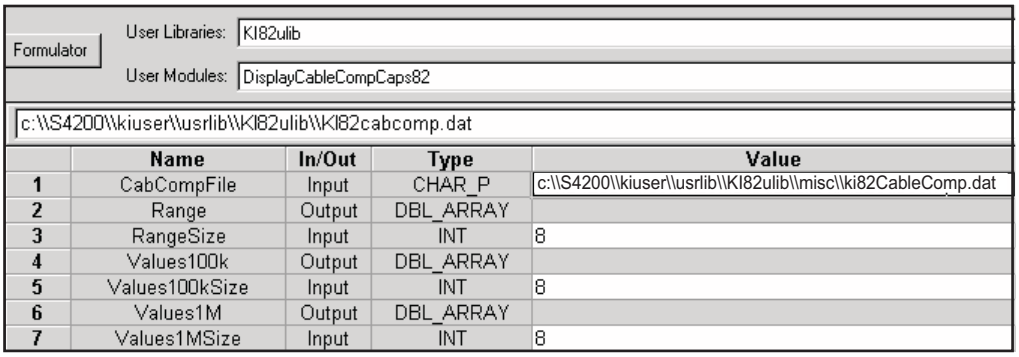
The default parameters for this user module are shown in Figure E-34. Line 1 specifies the file directory path where the capacitance values are saved. This file directory path must be the same as the one used by the **SaveCableCompCaps82** user module.

To prevent unpredictable results, the array size values for the Range, 100 k, and 1 M arrays (lines 3, 5, and 7) must be set to 8 as shown in Figure E-34.

User-entered parameters and returned outputs for this user module are explained in the User module description.

NOTE: *For details on the procedure to perform cable compensation see "Model 82 project plans, Cable compensation tests."*

Figure E-34
DisplayCableCompCap82 user module



	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	Range	Output	DBL_ARRAY	
3	RangeSize	Input	INT	8
4	Values100k	Output	DBL_ARRAY	
5	Values100kSize	Input	INT	8
6	Values1M	Output	DBL_ARRAY	
7	Values1MSize	Input	INT	8

User module description

The **DisplayCableCompCaps82** reads the nominal cable compensation values that are stored in the compensation file, and returns them to the calling function or in the case of KITE, to the UTM data sheet. The returned arrays are arranged in the following order:

Table E-8
DisplayCableCompCaps82 returned arrays

Range	100 kHz values	1 MHz values
2 e-12	2 pF low comp value	2 pF low comp value
2 e-12	2 pF high comp value	2 pF high comp value
20 e-12	20 pF low comp value	20 pF low comp value
20 e-12	20 pF high comp value	20 pF high comp value
200 e-12	200 pF low comp value	200 pF low comp value
200 e-12	200 pF high comp value	200 pF high comp value
2 e-9	2 nF low comp value	2 nF low comp value
2 e-9	2 nF high comp value	2 nF high comp value

SYNTAX:

status = DisplayCableCompCaps82(char *CabCompFile, double *Range, int RangeSize, double *Values100 k, int Values100 kSize, double *Values1M, int Values1MSize);

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, be sure to use two '\\' characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

```
C:\\calfiles\\590cal.dat
```

Values100 kSize,(int)
Values1MSize,
RangeSize The size of the Range, Values100 k, and Values1M arrays. THESE PARAMETERS MUST BE SET TO 8 TO AVOID UNPREDICTABLE RESULTS.

OUTPUTS:

Range	(double array) An 8 element array that receives the nominal range values.
Values100 k	(double array) An 8 element (fixed) array that receives the nominal capacitor values used to perform the cable compensation at the 100 kHz frequency.
Values1M	(double array) An 8 element (fixed) array that receives the nominal capacitor values used to perform the cable compensation at the 1 MHz frequency.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0	OK.
-10021	(COMP_FILE_NOT_EXIST) The specified compensation file does not exist.
-10022	(KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.
-10090	(GPIB_ERROR_OCCURRED) There was a GPIB communications error.
-10100	(INVAL_PARAM) An invalid parameter was specified.

QTsweep82 user module

Overview

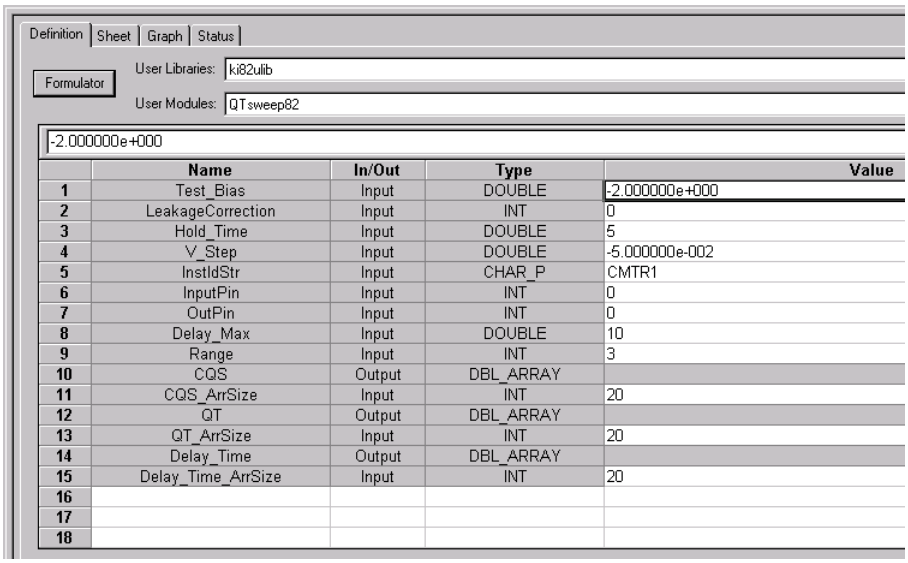
This user module uses the Model 595 to determine the equilibrium point for a device by measuring quasistatic capacitance using different delay times. Each quasistatic capacitance reading is calculated from charge measurements performed on every two steps of a voltage sweep. Leakage current at the end of each reading sample is also calculated ($i = \Delta Q/\Delta t$).

Figure E-35 shows the default parameters for the QTsweep82 user module. The QTsweep in Figure E-19 acquires 20 quasistatic capacitance readings. After the graph for quasistatic capacitance and leakage current vs. time is plotted, the optimum delay time for equilibrium can be determined.

NOTE: *For details on quasistatic measurements, see "Model 82 project plans, QTsweep."*

User-entered parameters and outputs for this user module are explained in the User module description.

Figure E-35
QTsweep82 user module



	Name	In/Out	Type	Value
1	Test_Bias	Input	DOUBLE	-2.000000e+000
2	LeakageCorrection	Input	INT	0
3	Hold_Time	Input	DOUBLE	5
4	V_Step	Input	DOUBLE	-5.000000e-002
5	InstIdStr	Input	CHAR_P	CMTR1
6	InputPin	Input	INT	0
7	OutPin	Input	INT	0
8	Delay_Max	Input	DOUBLE	10
9	Range	Input	INT	3
10	CQS	Output	DBL_ARRAY	
11	CQS_ArrSize	Input	INT	20
12	QT	Output	DBL_ARRAY	
13	QT_ArrSize	Input	INT	20
14	Delay_Time	Output	DBL_ARRAY	
15	Delay_Time_ArrSize	Input	INT	20
16				
17				
18				

User module description

The module measures quasistatic capacitance and leakage current as a function of delay time using the Model 595. It is used to determine the equilibrium condition.

NOTE: *For details on C-t measurements, see "Model 82 project plans, C-t sweep."*

SYNTAX:

status = QTsweep82(double Test_Bias, int LeakageCorrection, double Hold_time, double V-Step, char *InstIdStr, int InputPin, int OutPin, double Delay_Max, int Range, double *CQS, int, CQS_ArrSize, double *QT, int QT_ArrSize, double *Delay_time, int Delay_time_ArrSize);

PROCEDURE:

1. If desired, you will be prompted to open the circuit so an offset capacitance measurement can be made.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded. If not, instrument default compensation will be used.

A QT sweep is performed.

INPUTS:

Test_Bias (double) Bias at which a capacitance reading is taken; -120 to +120 (volts).

Hold_Time (double) Hold Time (in seconds) at the beginning of sweep; Minimum = 0, maximum = 200, default = 5.

V_Step (double) Step voltage size. Valid values: (+/-) 0, 0.01, 0.02, 0.05, 0.1 (volts).

InstIdStr (char *) The CMTR instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the KI5951's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the 5951 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72.

OutPin (int) The DUT pin to which the KI5951's output terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the 5951 output will be connected to the specified pin. Valid values for this parameter are -1 to 72.

Delay_Max (double) Maximum delay time in seconds; minimum = 1, maximum = 199.99, default = 10.

Range (int) The measurement range for the Model 595 to use. The valid range of range values is 1 to 3:

Table E-9
SaveCableCompCaps82 valid range values

Range value	Model 595 range
1	200 pF
2	2 nF
3	20nF

CQS_ArrSize (int) These array values *must* be equal to 20.

QT_ArrSize,

Delay_Time_ArrSize

OUTPUTS:

CQS	(double *) The measured array of quasistatic capacitance values.
QT	(double *) The measured array of leakage current Q/T.
Delay_Time	(double *) The array of Delay_Time used up to Delay_Max in logarithm scale.

RETURNED STATUS VALUES:

0	OK.
-10000	(INVAL_INST_ID) The specified instrument ID does not exist.
-10045	(KI82_NOT_IN_KCON) There is no CMTR defined in your system's configuration.
-10090	(GPIB_ERROR_OCCURED) A GPIB communications error occurred.
-10091	(GPIB_TIMEOUT) A time-out occurred during communications.
-10100	(INVAL_PARAM) An invalid parameter was specified.
-10101	(ARRAY_SIZE_TOO_SMALL) The specified value for CQS_ArrSize, QT_ArrSize, or Delay_Time_ArrSize was too small for the number of steps in the sweep.
-10102	(ERROR_PARSING) There was an error parsing the response.
-10104	(USER_CANCEL) The user CANCELED the correction procedure.

SaveCableCompCaps82 user module

Overview

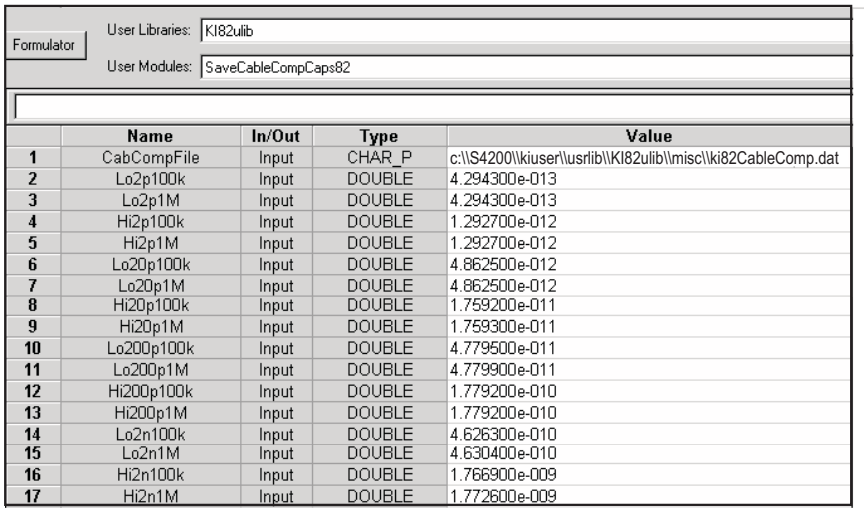
This user module is used for Model 590 cable compensation. The user enters precise capacitance source values. When this test is run, the capacitance source values are saved to a user-specified file. The user module to perform cable compensation (**CableCompensate82**) can then access the capacitance source values from this file.

The default parameter values for this user module are shown in Figure E-36. These are example low / high values that can be used for cable compensation. You must replace these values with the calibration values of the capacitance sources.

User-entered parameters and returned outputs for this user module are explained in the User module description.

NOTE: *For details on the procedure to perform cable compensation, see "Model 82 project plans, cable compensation tests."*

Figure E-36
SaveCableCompCaps82 user module



	Name	In/Out	Type	Value
1	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
2	Lo2p100k	Input	DOUBLE	4.294300e-013
3	Lo2p1M	Input	DOUBLE	4.294300e-013
4	Hi2p100k	Input	DOUBLE	1.292700e-012
5	Hi2p1M	Input	DOUBLE	1.292700e-012
6	Lo20p100k	Input	DOUBLE	4.862500e-012
7	Lo20p1M	Input	DOUBLE	4.862500e-012
8	Hi20p100k	Input	DOUBLE	1.759200e-011
9	Hi20p1M	Input	DOUBLE	1.759300e-011
10	Lo200p100k	Input	DOUBLE	4.779500e-011
11	Lo200p1M	Input	DOUBLE	4.779900e-011
12	Hi200p100k	Input	DOUBLE	1.779200e-010
13	Hi200p1M	Input	DOUBLE	1.779200e-010
14	Lo2n100k	Input	DOUBLE	4.626300e-010
15	Lo2n1M	Input	DOUBLE	4.630400e-010
16	Hi2n100k	Input	DOUBLE	1.766900e-009
17	Hi2n1M	Input	DOUBLE	1.772600e-009

User module description

This function saves the nominal values of the capacitors used to perform the Model 590 cable compensation procedure to the indicated file. If no cable compensation file exists, then this module will create one provided the user has the proper system permissions.

SYNTAX:

status = SaveCableCompCaps82(char *CabCompFile, double Lo2p100 k, double Lo2p1M, double Hi2p100 k, double Hi2p1M, double Lo20 p100 k, double Lo20 p1M, double Hi20 p100 k, double Hi20 p1M, double Lo200 p100 k, double Lo200 p1M, double Hi200 p100 k, double 200 p1M, double Lo2n100 k, double Lo2n1M, double Hi2n100 k, double Lo2n1M);

INPUTS:

CabCompFile (char *) The complete name and path for the cable compensation file. If this file does not exist or there is no path specified (null string), the default compensation parameters will be used. When entering the path, be sure to use two '\\' characters to separate each directory. For example, if your cable file is located in `C:\calfiles\590cal.dat`, you would enter the following:

`C:\\calfiles\\590cal.dat`

Lo2p100 k (double) The nominal value of the low range capacitor used to perform cable compensation for the 2 pF range and 100 kHz frequency. The valid range of inputs is 0 to 0.95 e-12 farads.

Lo2p1M (double) The nominal value of the low range capacitor used to perform cable compensation for the 2pF range and 1 MHz frequency. The valid range of inputs is 0 to 0.95 e-12 farads.

Hi2p100 k (double) The nominal value of the high range capacitor used to perform cable compensation for the 2pF range and 100 kHz frequency. The valid range of inputs is 1 e-12 to 2 e-12 farads.

Hi2p1M (double) The nominal value of the high range capacitor used to perform cable compensation for the 2 pF range and 1 MHz frequency. The valid range of inputs is 1 e-12 to 2 e-12 farads.

Lo20 p100 k	(double) The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency. The valid range of inputs is 0 to 9.5 e-12 farads.
Lo20 p1M	(double) The nominal value of the low range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency. The valid range of inputs is 0 to 9.5 e-12 farads.
Hi20 p100 k	(double) The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 100 kHz frequency. The valid range of inputs is 10 e-12 to 20 e-12 farads.
Hi20 p1M	(double) The nominal value of the high range capacitor used to perform cable compensation for the 20 pF range and 1 MHz frequency. The valid range of inputs is 10 e-12 to 20 e-12 farads.
Lo200 p100 k	(double) The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency. The valid range of inputs is 0 to 95 e-12 farads.
Lo200 p1M	(double) The nominal value of the low range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency. The valid range of inputs is 0 to 95 e-12 farads.
Hi200 p100 k	(double) The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 100 kHz frequency. The valid range of inputs is 100 e-12 to 200 e-12 farads.
Hi200 p1M	(double) The nominal value of the high range capacitor used to perform cable compensation for the 200 pF range and 1 MHz frequency. The valid range of inputs is 100 e-12 to 200 e-12 farads.
Lo2n100 k	(double) The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency. The valid range of inputs is 0 to 995 e-12 farads.
Lo2n1M	(double) The nominal value of the low range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency. The valid range of inputs is 0 to 995 e-12 farads.
Hi2n100 k	(double) The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 100 kHz frequency. The valid range of inputs is 1000 e-12 to 2000 e-12 farads.
Hi2n1M	(double) The nominal value of the high range capacitor used to perform cable compensation for the 2 nF range and 1 MHz frequency. The valid range of inputs is 1000 e-12 to 2000 e-12 farads.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0	OK.
-10000	(INVAL_INST_ID) An invalid instrument ID was specified. This generally means that there is no instrument with the specified ID in your configuration.
-10001	(INVAL_PIN_SPEC) An invalid DUT pin number was specified.
-10003	(NO_SWITCH_MATRIX) No switch matrix was found.
-10004	(NO_MATRIX_CARDS) No matrix cards were found.
-10020	(COMP_FILE_ACCESS_ERR) There was an error accessing the cable compensation file.
-10021	(COMP_FILE_NOT_EXIST) The specified compensation file does not exist.

-10022 (KI590_NOT_IN_KCON) There is no CMTR defined in your system's configuration.

SIMCVsweep82 user module

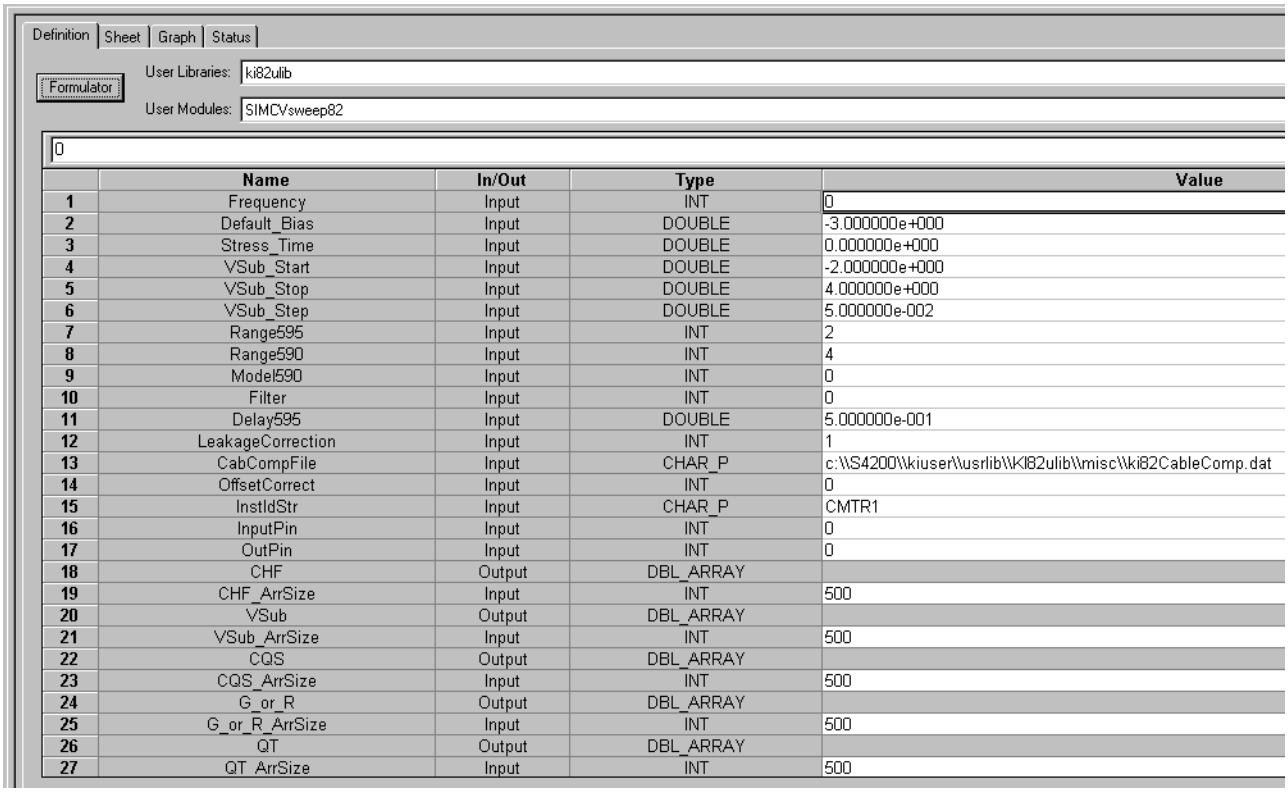
Overview

This user module uses the Models 590 and 595 to perform simultaneous C-V measurements. Figure E-37 shows the default parameters for the **SIMCVsweep82** user module. It will perform a staircase sweep from -3V to +3V in 20 mV steps as shown in Figure E-22.

User-entered parameters and outputs for this user module are explained in the User module description.

NOTE: *For details on quasistatic measurements, see "Model 82 project plans, Simultaneous C-V sweep."*

Figure E-37
SIMCVsweep82 user module



	Name	In/Out	Type	Value
1	Frequency	Input	INT	0
2	Default_Bias	Input	DOUBLE	-3.000000e+000
3	Stress_Time	Input	DOUBLE	0.000000e+000
4	VSub_Start	Input	DOUBLE	-2.000000e+000
5	VSub_Stop	Input	DOUBLE	4.000000e+000
6	VSub_Step	Input	DOUBLE	5.000000e-002
7	Range595	Input	INT	2
8	Range590	Input	INT	4
9	Model590	Input	INT	0
10	Filter	Input	INT	0
11	Delay595	Input	DOUBLE	5.000000e-001
12	LeakageCorrection	Input	INT	1
13	CabCompFile	Input	CHAR_P	c:\\S4200\\kiuser\\usrlib\\KI82ulib\\misc\\ki82CableComp.dat
14	OffsetCorrect	Input	INT	0
15	InstIdStr	Input	CHAR_P	CMTR1
16	InputPin	Input	INT	0
17	OutPin	Input	INT	0
18	CHF	Output	DBL_ARRAY	
19	CHF_ArrSize	Input	INT	500
20	VSub	Output	DBL_ARRAY	
21	VSub_ArrSize	Input	INT	500
22	CQS	Output	DBL_ARRAY	
23	CQS_ArrSize	Input	INT	500
24	G_or_R	Output	DBL_ARRAY	
25	G_or_R_ArrSize	Input	INT	500
26	QT	Output	DBL_ARRAY	
27	QT_ArrSize	Input	INT	500

User module description

The **SIMCVsweep82** routine performs a simultaneous capacitance vs. voltage (C-V) sweep using the Keithley Instruments Model 82 C-V System. If desired, an offset correction measurement is taken and the cable compensation can be used.

SYNTAX:

status = SIMCVsweep82(double Frequency, double Default_Bias, double Stress_Time, double VSub_Start, double VSub_Stop, double VSub_Step, int Range595, int Range590, int Model590, int Filter, double Delay595, int LeakageCorrection, char

*CabCompFile, int OffsetCorrect, char *InstIdStr, int InputPin, int OutPin, double *CHF, int CHF_ArrSize, double *VSub, int VSub_ArrSize, double *CQS, int CQS_ArrSize, double G_or_R, int G_or_R_ArrSize, double *QT, int QT_ArrSize);

PROCEDURE:

1. If desired, you will be prompted to open the circuit so that an offset capacitance measurement can be made.
2. If a cable compensation file is specified, the compensation information in that file for the selected range and frequency will be loaded.
3. A simultaneous C-V sweep is taken.

INPUTS:

Frequency (int) A variable that selects the measurement frequency to use for the Model 590. If set to 0, a frequency of 100 kHz will be used. If set to 1, the 1 MHz frequency will be selected.

Default_Bias (double) Default bias before and after voltage sweep; -100 to +100 (volts).

Stress_Time (double) Time for which default bias is stressed on the device before voltage sweep; 0 to 999 (seconds).

VSub_Start (double) Start Voltage on substrate; -120 to +120 (volts).

VSub_Stop (double) Stop Voltage on substrate; -120 to +120 (volts).

VSub_Step (double) Voltage step size.Valid values; (+/-) 0, 0.01, 0.02, 0.05 or 0.1 (volts).

Range595 (int) The measurement range for the Model 595 to use. The valid range of range values is 1 to 3:

Table E-10
SIMCsweep82 range595 valid range values

Range value	Model 595 range
1	200 pF
2	2 nF
3	20nF

Range590 (int) The measurement range for the Model 590 to use. The valid range of range values are 1 to 4:

Table E-11
SIMCsweep82 range590 valid range values

Range	100 kHz	1 MHz
1	2 pF / 2 uS	20 pF / 200 uS
2	20 pF / 20 uS	20 pF / 200 uS
3	200 pF / 200 uS	200 pF / 2mS
4	2 n F/ 2 mS	2 nF / 20 mS

Model590 (int) Which measurement model to use for high frequency measurements. Entering 0 for this parameter will select the parallel model, while 1 will select the series model.

Filter (int) Enable/disable the digital filter. Valid value from 0 to 3:

Table E-12
SIMCsweep82 filter valid values

Filter	Effect
0	1 reading
1	3 readings
2	9 readings
3	24 readings

Delay595 (double) Delay time for Model 595. Default value is 0.07 sec. Maximum is 199 sec.

LeakageCorrection (int) Enable or disable leakage current correction of Model 595; 0 = disable and 1 = enable.

CabCompFile (char *) The complete name and path of the cable compensation file. If this file does not exist, or this string is null, then cable compensation will not be used. The path that you specify must exist. (When entering the path information, be sure to use two '\\' characters to separate each directory level. For example, if your cable compensation file is located in file C:\calfiles\590cal.dat, you would enter C:\\calfiles\\590cal.dat).

OffsetCorrect (int) A flag indicating whether to perform an offset correction measurement. If OffsetCorrect is 0, then no offset measurement will be taken. If OffsetCorrect is 1, then an offset measurement will be made.

InstIdStr (char *) The CMTR 82 instrument ID. This can be CMTR1 through CMTR4, depending on your system's configuration.

InputPin (int) The DUT pin to which the Model 5951's input terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 5951 input will be connected to this DUT pin. Valid values for this parameter are -1 to 72.

OutPin (int) The DUT pin to which the Model 5951's output terminal will be attached. If a value of less than 1 is specified, no switch matrix connections will be made. Otherwise, the Model 5951 output will be connected to the specified pin. Valid values for this parameter are -1 to 72.

CHF_ArrSize, V_ArrSize, CQS_ArrSize, G_or_R_ArrSize, QT_ArrSize (int) These values *must* be set equal to a value that, at minimum, is equal to the number of voltage steps in the sweep, or:
value = ((VSub_Stop - VSub_Start) / VSub_Step -1)

OUTPUTS:

CHF (double *) The measured array of high frequency capacitance values.

VSub (double *) The array of bias voltages used.

CQS (double *) The measured array of quasistatic capacitance values.

G_or_R (double *) The array of measured conductance or resistance values.

QT (double) The array of Q/T from Model 595 output for each measurement step.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0 OK.

-10000	(INVAL_INST_ID) The specified instrument ID does not exist.
-10020	(COMP_FILE_ACCESS_ERR) There was an error accessing the specified cable compensation file.
-10021	(COMP_FILE_NOT_EXIST) The specified compensation file does not exist.
-10023	(KI590_MEAS_ERROR) A measurement error occurred.
-10090	(GPIB_ERROR_OCCURED) A GPIB communications error occurred.
-10091	(GPIB_TIMEOUT) A time-out occurred during communications.
-10100	(INVAL_PARAM) An invalid parameter was specified.
-10101	(ARRAY_SIZE_TOO_SMALL) The specified value for CHF_ArrSize, G_or_R_ArrSize, V_ArrSize, CQS_ArrSize or QT_ArrSize was too small for the number of steps in the sweep.
-10102	(ERROR_PARSING) There was an error parsing the response.
-10104	(USER_CANCEL) The user CANCELED the correction procedure.
-10045	(KI82_NOT_IN_KCON) KI82 is not in KCON.

Simultaneous C-V analysis

This section discusses the theory and techniques used in the various Keithley Instruments Simultaneous C-V libraries. For more detailed discussions, refer to the references at the end of the chapter.

Analysis methods

Basic simultaneous C-V curves

Figure E-38 and Figure E-39 show fundamental C-V curves for p-type and n-type materials respectively. Both high-frequency and quasistatic curves are shown in these figures. Note that the high-frequency curves are highly asymmetrical, while the quasistatic curves are almost symmetrical. Accumulation, depletion, and inversion regions are also shown on the curves. The gate-biasing polarity and high-frequency curve shape can be used to determine device type, as discussed below.

Figure E-38
C-V characteristics of p-type material

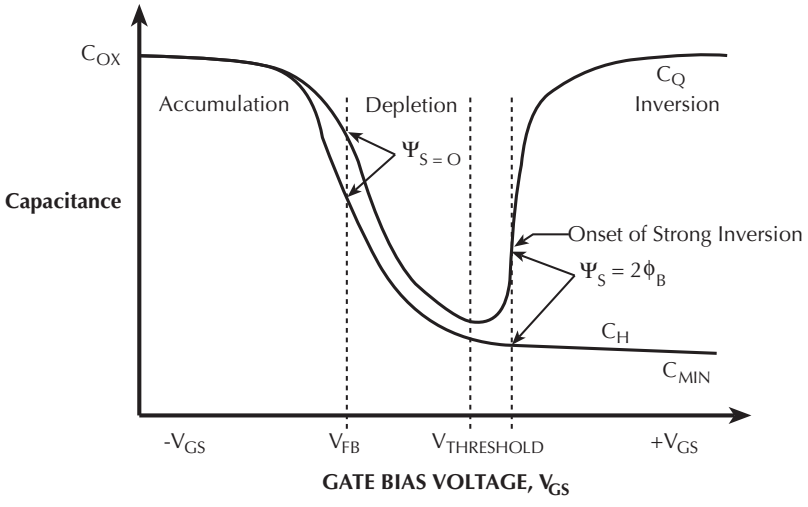
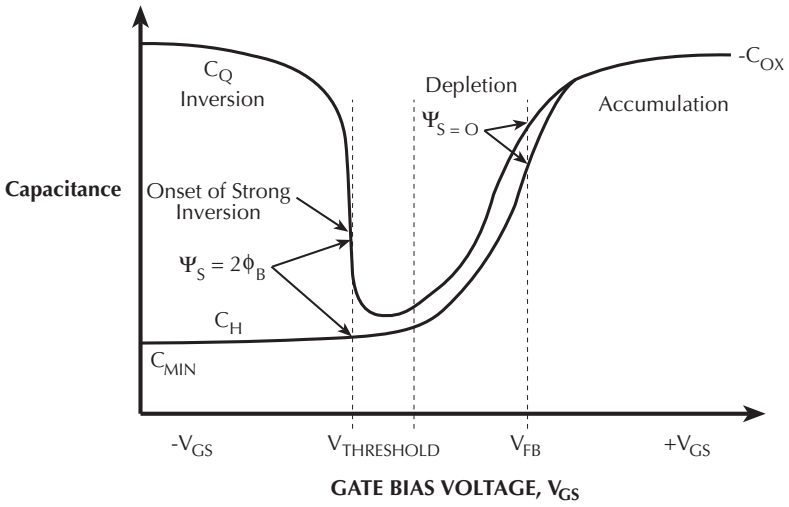


Figure E-39
C-V characteristics of n-type material



Basic device parameters

Determining device type

The semiconductor conductivity type (p or n dopant ions) can be determined from the relative shape of the C-V curves (see Figure E-38 and Figure E-39). The high-frequency curve gives a better indication than the quasistatic curve because of its highly asymmetrical nature. Note that the C-V curve moves from the accumulation to the inversion region as gate voltage, V_{GS}, becomes more positive for p-type materials, but the curve moves from accumulation to inversion as V_{GS} becomes more negative with n-type materials (Nicollian and Brews 372-374).

- If C_H is greater when V_{GS} is negative than when V_{GS} is positive, the substrate material is p-type.
- If, on the other hand, C_H is greater with positive V_{GS} than with negative V_{GS}, the substrate is n-type.
- The end of the curve where C_H is greater is the accumulation region, while the opposite end of the curve is the inversion

Oxide capacitance, thickness and gate area

The oxide capacitance, C_{ox}, is the high-frequency capacitance with the device biased in strong accumulation. Oxide thickness is calculated from C_{ox} and gate area as follows:

$$t_{OX} = \frac{A\,\varepsilon_{OX}}{(1\times10^{-19})C_{OX}} \qquad (1)$$

Where:

t_{ox} = oxide thickness (nm)
A = gate area (cm^2)

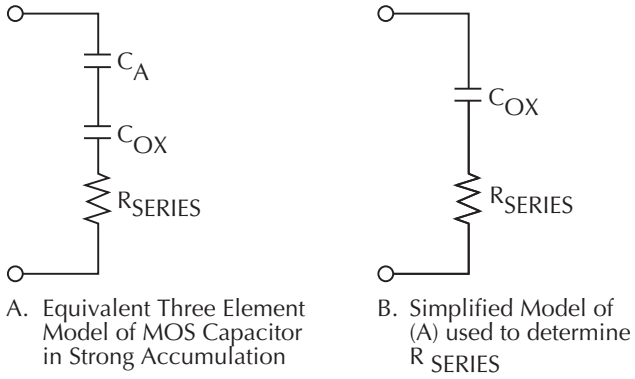
ε_{ox} = permittivity of oxide material (F/cm)
C_{ox} = oxide capacitance (pF)

The above equation can be easily rearranged to calculate gate area if the oxide thickness is known. Note that ε_{OX} and other constants are initialized for use with silicon substrate, silicondioxide insulator, and aluminum gate material but may be changed for other materials (see the Model 4200-SCS Reference Manual for details on modifying constants).

Series resistance

The series resistance, R_{SERIES} is an error term that can cause measurement and analysis errors unless this series resistance error factor is taken into account. Without series compensation, capacitance can be lower than normal, and C-V curves can be distorted. The software compensates for series resistance using the simplified three-element model shown in Figure E-40. In this model, C_{OX} is, of course, the oxide capacitance while C_A is the capacitance of the accumulation layer. The series resistance is represented by R_{SERIES}.

Figure E-40
Simplified model to determine series resistance



A. Equivalent Three Element Model of MOS Capacitor in Strong Accumulation

B. Simplified Model of (A) used to determine R_{SERIES}

From Nicollian and Brews 224, the correction capacitance, C_C, and corrected conductance, G_C, are calculated as follows:

$$C_C = \frac{(G_M^2+\omega^2 C_M^2)C_M}{a^2+\omega^2 C_M^2} \quad (2)$$

and,

$$G_C = \frac{(G_M^2+\omega^2 C_M^2)a}{a^2+\omega^2 C_M^2} \quad (3)$$

Where:

$a = G_M - (G^2_M + \omega^2 C^2_M) R_{SERIES}$
C_c = series resistance compensated parallel model capacitance
C_M = measured parallel model capacitance
G_C = series resistance compensated conductance
G_M = measured conductance
R_{SERIES} = series resistance

Gain and offset

Gain and offset can be applied to C_Q and C_H data to allow for curve alignment or to compensate for measurement errors. A gain factor is a multiplier that is applied to all elements of C_Q or C_H

array data before plotting or graphics array calculation. Offset is a constant value added to or subtracted from all C_Q and C_H data before plotting or array calculation.

For example, assume that you compare the C_Q and Cn values at reading #3, and you find that C_Q is 2.3pF less than Cn. If you then add an offset of +2.3pF to C_Q, the C_Q and C_H values at reading #3 will then be the same, and the C_Q and C_H curves will be aligned at that point.

Gain and offset values do not affect raw C_Q and C_H values stored in the data file, but the gain and offset values will be stored in the data file so compensated curves can easily be regenerated at a later date.

Flatband capacitance and flatband voltage

The Model 82 uses the flatband capacitance method of finding flatband voltage, V_{FB}. The Debye length is used to calculate the ideal value of flatband capacitance, C_{FB}. Once the value of C_{FB} is known, the value of V_{FB} is interpolated from the closest V_G values (Nicollian and Brews 487-488).

The method used is invalid when interface trap density becomes very large (1012-1013 and greater). However, this algorithm should give satisfactory results for most users. Those who are dealing with high values of D_{IT} should consult the appropriate literature for a more appropriate method.

Based on doping, the calculation of C_{FB} uses N at 90% W_{MAX}, or user-supplied N_A (bulk doping for p-type, acceptors) or N_D (bulk doping for n-type, donors).

C_{FB} is calculated as follows:

$$C_{FB} = \frac{C_{OX}\,\varepsilon_S A/(1\text{x}10^{-4})(\lambda)}{(1\times10^{-12})(C_{OX}) + \varepsilon_S A / (1\times10^{-4})(\lambda)} \qquad (4)$$

Where:

C_{FB} = flatband capacitance (pF)
C_{ox} = oxide capacitance (pF)
ε_S = permittivity of substrate material (F/cm)
A = gate area (cm^2)
1×10^{-4} = units conversion for λ
1×10^{-12} = units conversion for C_{ox}

And λ = extrinsic Debye length =

$$(1\times10^{4})\left(\frac{\varepsilon_S kT}{q^2 N_X}\right)^{1/2} \qquad (5)$$

Where:

kT = thermal energy at room temperature ($4{,}046 \times 10^{-21}$ J)
q = electron charge (1.60219×10^{-19} coul.)
N_x = N at 90% W_{MAX}, or N_A, or N_D when input by the user.

N at 90% W_{MAX} is chosen to represent bulk doping.

Threshold voltage

The threshold voltage, V_{TH}, is the point on the C-V curve where the surface potential ψ_S, equals twice the bulk potential, ϕ_B. This point on the curve corresponds to the onset of strong inversion. For an enhancement mode MOSFET, V_{TH} corresponds to the point where the device begins to conduct.

V_{TH} is calculated as follows:

$$V_{TH} = \left[\pm \frac{A}{10^{12} C_{OX}} \sqrt{4\, \varepsilon_S\, q |N_{BULK}| |\phi_B| + 2|\phi_B|} \right] + V_{FB} \quad (6)$$

Where:

V_{TH} = threshold voltage (V)
A = gate area (cm^2)
C_{ox} = oxide capacitance (pF)

10^{12} = units multiplier
ε_S = permittivity of substrate material
q = electron charge (1.60219×10^{-19} coul.)
N_{BULK} = bulk doping (cm^{-3})
ϕ_B = bulk potential (V)
V_{FB} = flatband voltage (V)

Metal semiconductor work function difference

The metal semiconductor work function difference, W_{MS}, is commonly referred to as the work function. It contributes to the shift in V_{FB} from the ideal zero value, along with the effective oxide charge (Nicollian and Brews 462-477; Sze 395402). The work function represents the difference in work necessary to remove an electron from the gate and from the substrate, and it is derived as follows:

$$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B \right] \quad (7)$$

Where:

W_M = metal work function (V)
W_S = substrate material work function (electron affinity) (V)
E_G = substrate material bandgap (V)
ϕ_B = bulk potential (V)

For silicon, silicon dioxide, and aluminum:

$$W_{MS} = 4.1 - \left[4.15 + \frac{1.12}{2} - \phi_B \right] \quad (8)$$

So that,

$$W_{MS} = -0.61 + \phi_B \quad (9)$$

$$W_{MS} = -0.61 - \left(\frac{kT}{q} \right) \ln\left(\frac{N_{BULK}}{n_I} \right) (DopeType) \quad (10)$$

Where, DopeType is +l for p-type materials, and -1 for n-type materials. For example, for a MOS capacitor with an aluminum gate and p-type silicon ($N_{BULK} = 10^{16} cm^{-3}$), W_{MS} = -0.95 V. Also, for the same gate and n-type silicon ($N_{BULK} = 10^{16} cm^{-3}$), W_{MS} = -0.27V.

Effective oxide charge

The effective oxide charge, Q_{EFF}, represents the sum of oxide fixed charge, Q_F, mobile ionic charge, Q_M and oxide trapped charge, Q_{OT}. Q_{EFF} is distinguished from interface trapped charge, Q_{IT}, in that Q_{IT} varies with gate bias and $Q_{EFF} = Q_F + Q_M + Q_{OT}$ does not (Nicollian and Brews

424-429, Sze 390-395). Simple measurements of oxide charge using C-V measurements do not distinguish the three components of Q_{EFF}.

These three components can be distinguished from one another by temperature cycling, as discussed in Nicollian and Brews, 429, Fig. 10.2. Also, since the charge profile in the oxide is not known, the quantity, Q_{EFF} should be used as a relative, not absolute measure of charge. It assumes that the charge is located in a sheet at the silicon-silicon dioxide interface. From Nicollian and Brews, Eq. 10. 10, we have:

$$V_{FB} - W_{MS} = -\frac{Q_{EFF}}{C_{OX}} \quad (11)$$

Note that C_{OX} here is per unit of area. So that,

$$Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A} \quad (12)$$

However, since C_{OX} is in F, we must convert to pF by multiplying by 10^{-12} as follows:

$$Q_{EFF} = 10^{-12}\frac{C_{OX}(W_{MS} - V_{FB})}{A} \quad (13)$$

Where:

Q_{EFF} = effective charge (coul/cm^2)
C_{OX} = oxide capacitance (pF)
W_{MS} = metal semiconductor work function (V)
A = gate area (cm^2)

For example, assume a 0.01cm^2 50 pF capacitor with a flatband voltage of -5.95 V, and a p-type $N_{BULK} = 10^{16}$cm^{-3} (resulting in W_{MS} = -0.95 V). In this case, $Q_{EFF} = 2.5 \times 10^{-4}$ coul/cm^2.

The effective oxide charge concentration, N_{EFF}, is computed from effective oxide charge and electron charge as follows:

$$N_{EFF} = \frac{Q_{EFF}}{q} \quad (14)$$

Where:

N_{EFF} = effective concentration of oxide charge (Units of charge/cm^2)
Q_{EFF} = effective oxide charge (coul./cm^2)
q = electron charge (1.60219×10^{-19} coul.)

For example, with an effective oxide charge of 2.5×10^{-8} coul/cm^2, the effective oxide charge concentration is:

$$N_{EFF} = \frac{2.5 \times 10^{-8}}{1.60219 \times 10^{-19}} \quad (15)$$

$$N_{EFF} = 1.56 \times 10^{11} units / cm^2 \quad (16)$$

Doping profile

Depletion depth vs. gate voltage (VGS)

The Model 82 computes the depletion depth, w, from the high-frequency capacitance and oxide capacitance at each measured value of V_{GS} (Nicollian and Brews 386). In order to graph this function, the program computes each w element of the calculated data array as shown below:

$$w = A\,\varepsilon_S\left(\frac{1}{C_H} - \frac{1}{C_{OX}}\right) \quad (17)$$

Where:

w = depth (µm)
ε_S = permittivity of substrate material
C_H = high-frequency capacitance (pF)
C_{OX} = oxide capacitance (pF)
A = gate area (cm^2)

$1/C_H^2$ vs. gate voltage

A $1/C^2$ graph can yield important information about doping profile. N is related to the reciprocal of the slope of the $1/C^2$ vs. V_{GS} curve, and the V intercept point is equal to the flatband voltage caused by surface charge and metal-semiconductor work function (Nicollian and Brews 385).

Doping concentration vs. depth

The doping profile of the device is derived from the C-V curve based on the definition of the differential capacitance (measured by the Models 590 and 595) as the differential change in depletion region charge produced by a differential change in gate voltage (Nicollian and Brews 380-389).

The standard N vs. w analysis discussed here does not compensate for the onset of accumulation, and it is accurate only in depletion. This method becomes inaccurate when the depth is less than two Debye lengths.

In order to correct for errors caused by interface traps, the error term $(1\text{-}C_Q/C_{OX})/1\text{-}C_H/C_{OX})$ is included in the calculations as follows:

$$N = \frac{(-2\times10^{-24})[(1-C_Q/C_{OX})/(1-C_H/C_{OX})]}{A^2 q\,\varepsilon_S}\left[\frac{d}{dV_{GS}}\left(\frac{1}{C_H^2}\right)\right]^{-1} \quad (18)$$

Where:

N = doping concentration (cm^{-3})
C_Q = quasistatic capacitance (pF)
C_{OX} = oxide capacitance (pF)
$(1\text{-}C_Q/C_{OX})/1\text{-}C_H/C_{OX})$ = voltage stretchout term
C_H = high-frequency capacitance (pF)
A = gate area (cm^2)
q = electron charge (1.60219×10^{-19} coul.)
ε_S = permittivity of substrate material
1×10^{-24} = units conversion factor

Interface trap density

Band bending (ψ_s) vs. gate voltage

As a preliminary step, surface potential (ψ_S -ψ_0) vs. V_{GS} is calculated with the results placed in the ψ_S column of the array. Surface potential is calculated as follows:

$$(\Psi_S - \Psi_0) = \sum_{V_{GS}\#1}^{V_{GS}Last} (1 - C_Q / C_{OX})(2V_{STEP}) \quad (19)$$

Where:

(ψ_S -ψ_0) = surface potential (V)
C_Q = quasistatic capacitance (pF)
C_{OX} = oxide capacitance (pF)
V_{STEP} = step voltage (V)
V_{GS} = gate-substrate voltage (V)

Note that the (ψ_S -ψ_0) value is accumulated as the column is built, from the first row of the array (V_{GS} #l) to the last array row (V_{GS} last). The number of rows will, of course, depend on the number of readings in the sweep, which is determined by the Start, Stop, and Step voltages.

Once (ψ_S -ψ_0) values are stored in the array, the value of (ψ_S -ψ_0) at the flatband voltage is used as a reference point and is set zero by subtracting that value from each entry in the (ψ_S -ψ_0) column, changing each element in the column to ψ_S.

Interface trap capacitance CIT and density DIT

Interface trap density is calculated from C_{IT} as shown below (Nicollian and Brews 322).

$$C_{IT} = \left[\left(\frac{1}{C_Q} - \frac{1}{C_{OX}}\right) - \left(\frac{1}{C_H} - \frac{1}{C_{OX}}\right)\right]^{-1} \quad (20)$$

And,

$$D_{IT} = \frac{(1\times10^{-12})C_{IT}}{A} \quad (21)$$

Where:

C_{IT} = interface trap capacitance (pF)
D_{IT} = interface trap density (cm^{-2} eV^{-1})
C_Q = quasistatic capacitance (pF)
C_H = high-frequency capacitance (pF)
C_{OX} = oxide capacitance (pF)
A = gate area (cm^2)
q = electron charge (1.60219×10^{-19} coul.)
1×10^{-12} = units conversion for C_{IT}

Mobile ion charge concentration

Mobile ion monitoring with triangular voltage sweep (STVS) method

STVS is a new technique developed by Keithley Instruments to monitor mobile ion charge in MOS structures. Compared with other mobile ion monitoring techniques, such as the BTS and flatband shift methods, it offers faster and more accurate measurement. STVS measures ionic current instead of voltage shift. It has the ability to identify species, and it eliminates the need for

temperature cycling of the device under test (DUT). The STVS method has proven to be effective in monitoring mobile ion charge in dielectrics to levels down to $10^9 cm^{-3}$.

The STVS library can perform the corresponding mobile ion charge analysis. It has a built-in correction algorithm to eliminate the problems associated with leakage current. Many parameters, including mobile ion charge concentration, can be extracted from this measurement.

The STVS method improves on the conventional TVS method (discussed below) by measuring both C_Q and C_H and then computing mobile ion charge concentration as follows:

$$N_M = \frac{\sum_{-V_{GS}}^{+V_{GS}} (C_Q - C_H)\Delta V_{GS}}{q} \quad (22)$$

Where:

N_M = mobile ion density ($1/cm^3$)
V_{GS} = gate-substrate voltage (V)
ΔV_{GS} = change in gate-substrate voltage (step voltage) (V)
C_Q = quasistatic capacitance measured by Model 595 (F)
C_H = high-frequency capacitance measured by Model 590 (F)
q = electron charge (coul.)

Flatband voltage shift method

The primary method for measuring oxide charge density is the flatband voltage shift or temperature-bias stress method (Snow et al). In this case, two high-frequency C-V curves are measured, both at room temperature. Between the two curves, the device is biased with a voltage at 200-300° to drift mobile ions across the oxide. The flatband voltage differential between the two curves is then calculated, from which charge density can be determined.

From Nicollian and Brews (426, Eq. 10.9 and IO. lo), we have:

$$V_{FB} - W_{MS} = \frac{\bar{x} Q_O}{\varepsilon_{OX}} = \frac{\bar{x} Q_O}{X_O C_{OX}} \quad (23)$$

Where:

$\bar{x}Q_O$ = the first moment of the charge distribution
$\bar{x}$ = charge centroid
W_{MS} = metal semiconductor work function (constant)
ε_{OX} = oxide dielectric constant
X_O = oxide thickness
C_{OX} = oxide capacitance

So that,

$$\Delta V_{FB} = \Delta(V_{FB} - W_{MS}) \quad (24)$$

$$\Delta V_{FB} = \Delta \frac{\bar{x} Q_O}{\varepsilon_{OX}} \quad (25)$$

$$\Delta V_{FB} = \frac{Q_O}{C_{OX}} \Delta \frac{\bar{x}}{X_O} \quad (26)$$

For the common case of thermally grown oxide, $\bar{x}$ (before) = X_O and $\bar{x}$ (after) = 0, so that

$$\Delta V_{FB} = \frac{-Q_O}{C_{OX}} \quad (27)$$

Where Q_O is the effective charge. Divide Q_O by the gate area to obtain mobile ion charge density per unit area.

Simultaneous Triangular Voltage Sweep method for determining mobile oxide charges

The Simultaneous Triangular Voltage Sweep (STVS) method is very useful in determining the amount and type of mobile carriers that are in the oxide. This method uses a triangular voltage ramp applied to the gate of the device. The Model 595 applies a similar voltage ramp during its measurement. The Model 595 measures the ionic displacement current, while the device is at an elevated temperature. Elevating the temperature to approximately 300°C causes the high frequency curve to rise in inversion until it is similar to the quasistatic curve. If there are no mobile charges, the quasistatic curve remains approximately the same shape, except the depletion capacitance starts to approach the oxide capacitance. If mobile charges exist, a capacitance spike will appear on the quasistatic C-V curve when the mobile charges move from one side of the oxide to the other.

The quasistatic curve will peak during the movement of the mobile charge. Calculation of the mobile charge involves taking the difference in the high frequency and quasistatic capacitance and multiplying by the change in V_{GS} as shown in the following:

$$N_m = \frac{+V_{GS}}{-V_{GS}}(C_q - C_b)V_{GS}/(qA)$$

Where:

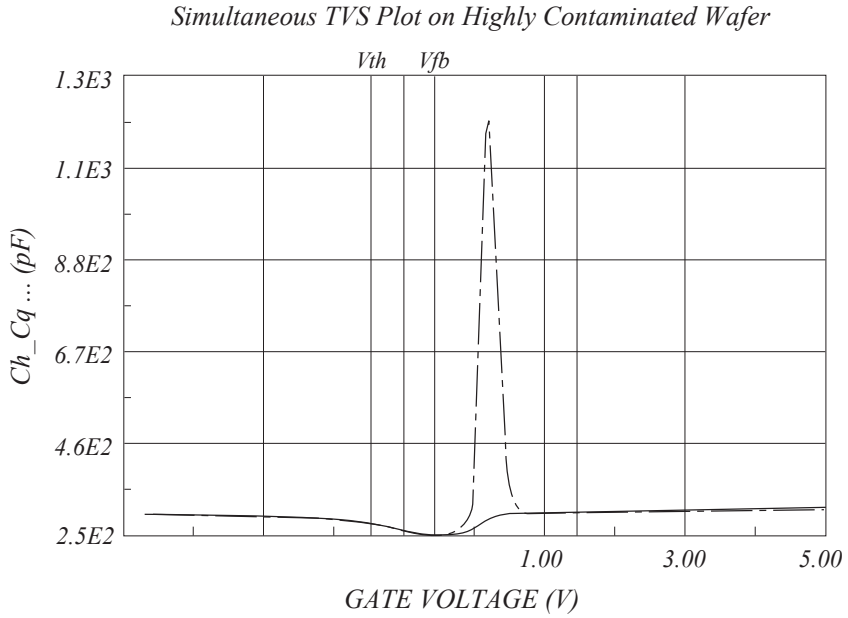
N_m = mobile ion concentration (cm^{-2})
$+V_{GS}$ = gate-substance voltage (V)
$-V_{GS}$ = change in gate-substrate voltage (V)
C_q = quasistatic capacitance at given V_{GS} (pF)
C_b = high frequency capacitance at given V_{GS} (capacitance without mobile charges) (pF)
q = electron charge = $1.60219X10^{-19}C$
A = area of gate capacitor (cm^2)

Figure E-41 demonstrates what a contaminated oxide should produce for a STVS curve.

This method has four advantages over the BTS method:

1. It determines the mobile charges without interference from the interface trap charges.
2. It can determine the type of ion (sodium or potassium) that is contaminating the oxide, because the peak in gate current for different ions occurs at different gate biases.
3. It provides measurements an order of magnitude more sensitive than bias temperature stress BTS.
4. It is faster than the BTS method, since the device only needs heating once and the calculation needs only one curve.

Figure E-41
Simultaneous TVS plot on a highly contaminated wafer



Calculation of the mobile charge concentration could come from the measured V_{GS}, C_q, and C_h data. Alternatively, one can calculate the concentration graphically from the displayed simultaneous C-V curves.

Generation velocity and generation lifetime (Zerbst plot)

Zerbst plot

Zerbst analysis requires two types of data: C-V and C-t. Important data taken from the C-V measurement includes C_{OX}, C_{MIN}, and doping concentration (N_{AVG} and N_{BULK}). The results of the C-V analysis are integrated with data taken during a C-t measurement to compute generation velocity and generation lifetime of electron-hole pairs. These two parameters are computed from the slope and y-axis intercept of the graph of G/n_I vs. $w\text{-}w_F$ as outlined below.

G/n_I computation

$$G / n_I = -\varepsilon_S A N_{AVG} C_{OX} \bullet \left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_I t_{\text{int}}} \right] \bullet \left(\frac{1\times10^{12}}{2} \right) \quad (30)$$

Where:

G = generation rate (s^{-1})
ε_S = permittivity of semiconductor (F/cm)
A = gate area (cm^2)
N_{AVG} = average doping concentration (cm^{-3})
C_{OX} = oxide (maximum) capacitance (pF)
$C_{t(i+1)}$ = (i+1) value of measured C-t capacitance (pF)
$C_{t(i-1)}$ = (i-1) value of measured C-t capacitance (pF)
n_I = intrinsic carrier concentration (cm^{-3})

t_{int} = time interval between C-t measurements (s)
i = [2, #Rdgs-1]

w - w_F computation

$$w - w_F = 1\times10^{12}\,\varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}}\right) - w_F \quad (31)$$

$$w_F = 1\times10^{12}\,\varepsilon_S A\left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}}\right) \quad (32)$$

Where:

w = depletion depth (cm)
w_F = equilibrium inversion depth (cm)
ε_S = permittivity of semiconductor (F/cm)
A = gate area (cm^2)
C_{ti} = i(th) value of measured C-t capacitance (pF)
C_{MIN} = equilibrium minimum capacitance (pF)

Determining generation velocity and generation lifetime

The generation lifetime, τ_G is equal to the reciprocal of the slope of the linear portion of the Zerbst plot, while the generation velocity, s, is the y-axis (G/n_I) intercept of the same linear section of the Zerbst plot.

Constants, symbols, and equations used for analysis

In order to perform correct analysis, it may be necessary for you to verify or modify the analysis constants to suit your particular device. Before making measurements, it is strongly recommended that you verify that constants are correct to ensure that your analysis is performed correctly. Otherwise, your analysis results will be meaningless.

Default material constants

Table E-13 lists default material constants, values, descriptions, and symbols.

Table E-13
Default material constants

Symbol	Description	Default value
q	Electron charge (Coul.)	1.60219×10^{-19} Coul.
k	Boltzmann's constant (J/°K)	1.38066×10^{-23} J/°K
T	Test temperature (°K)	293°K
ε_{OX}	Permittivity of oxide (F/cm)	3.4×10^{-13} F/cm
ε_S	Semiconductor permittivity (F/cm)	1.04×10^{-12} F/cm
E_G	Semiconductor energy gap (eV)	1.12 eV
n_I	Intrinsic carrier concentration (I/cm^3)	1.45×10^{10} cm^{-3}
W_{MS}	Metal work function (V)	4.1 V
	Electron affinity (V)	4.15 V

Data symbols

Table E-14 summarizes data symbols in the library along with a description of each symbol.

Table E-14
Data symbols

Symbol	Description	Units
A	Device gate area.	cm^2
C_{FB}	Flatband capacitance, corresponding to no band bending.	pF
C_H	High-frequency capacitance, as measured by the Model 590 at either 100 kHz or 1 MHz.	pF
C_{HADJ}	The high-frequency capacitance that is adjusted according to gain and offset values. C_{HADJ} is the value that is actually plotted and printed.	pF
C_Q	Quasistatic capacitance as measured by Model 590.	pF
C_{QADJ}	The quasistatic capacitance that is adjusted according to gain and offset values. C_{QADJ} is the value that is actually plotted and printed.	pF
C_Q'	Interpolated value of C_Q set to correspond to the quasistatic capacitance at V.	pF
C_{MIN}	Minimum high-frequency capacitance in inversion.	pF
C_{OX}	Oxide capacitance, usually set to the maximum C_H in accumulation.	pF
D_{rr}	Density or concentration of interface states.	$1/cm^2/eV$
E_C	Energy of conduction band edge (valence band is E_V).	eV
E_T	Interface trap energy.	eV
G	High-frequency conductance, as measured by the Model 590 at either 100 kHz or 1 MHz.	S
N_A	Bulk doping for p-type (acceptors).	$1 / cm^3$
N_D	Bulk doping for n-type (donors).	$1 / cm^3$
N_{AVG}	Average doping concentration.	$1 / cm^3$
N_{BULK}	Bulk doping concentration.	$1 / cm^3$
N_{EFF}	Effective oxide charge concentration.	$1 / cm^2$
N(90% W_{MAX})	Doping corresponding to 90% maximum w profile (approximates doping in the bulk).	$1 / cm^3$
N_M	Mobile ion concentration in the oxide.	$1 / cm^3$
Q_{EFF}	Effective oxide charge.	coul / cm^2
Q / t	Current measured by the Model 595 at the end of each capacitance measurement with the unit in the capacitance function.	A
R_{SERIES}	Series resistance.	¾
t_{OX}	Oxide thickness.	nm
V_{GS}	Gate voltage. More specifically, the voltage at the gate with respect to the substrate.	V
V_{FB}	Flatband voltage, or the value of V_{GS} that results in C_{FB}.	V
V_H	Voltage reading sent by Model 590 with matching C_H and G.	V
V_{TH}	The point where the surface potential, ψ_S, is equal to twice the bulk potential, ϕ_B.	V
w	Depletion depth or thickness. Silicon under the gate is depleted of minority carriers in inversion and depletion.	µm
ψ_S	Silicon surface potential as a function of V_{GS}. More precisely, this value represents band bending and is related to surface potential via the bulk potential.	V
ψ_0	Offset in ψ_S due to calculation method and V_0.	V
ϕ_B	Silicon bulk potential.	V
l	Extrinsic Debye length.	m

Summary of analysis equations

Table E-15 summarizes analysis equations used by the Model 82 software.

Table E-15
Analysis equations

Analysis function	Equation
Band bending	$(\Psi_S - \Psi_0) = \sum_{V_{GS}\#1}^{V_{GS}Last} (1 - C_Q / C_{OX})(2V_{STEP})$
Depletion depth	$w = A\,\varepsilon_S \left(\frac{1}{C_H} - \frac{1}{C_{OX}} \right)$
Doping concentration	$N = \frac{(-2\times10^{-24})[(1 - C_Q / C_{OX})/(1 - C_H / C_{OX})]}{A^2 q\,\varepsilon_S} \left[\frac{d}{dV_{GS}} \left(\frac{1}{C_H^2} \right) \right]^{-1}$
Effective oxide charge	$Q_{EFF} = \frac{C_{OX}(W_{MS} - V_{FB})}{A}$
Effective charge concentration	$N_{EFF} = \frac{Q_{EFF}}{q}$
Flatband capacitance	$C_{FB} = \frac{C_{OX}\,\varepsilon_S A / (1\text{x}10^{-4})(\lambda)}{(1\times10^{-12})(C_{OX}) + \varepsilon_S A / (1\times10^{-4})(\lambda)}$ Where: $\lambda = (1\times10^{4}) \left(\frac{\varepsilon_S kT}{q^2 N_X} \right)^{1/2}$ And N_X = N at 90% W_{MAX}, N_A, or N_D.
Flatband voltage shift	$V_{FB} - W_{MS} = \frac{\overline{x} Q_O}{\varepsilon_{OX}} = \frac{\overline{x} Q_O}{X_O C_{OX}}$ $\Delta V_{FB} = \frac{-Q_O}{C_{OX}}$

Table E-15 (continued)
Analysis equations

Analysis function	Equation
Interface trap capacitance	$C_{IT} = \left[\left(\frac{1}{C_Q} - \frac{1}{C_{OX}}\right) - \left(\frac{1}{C_H} - \frac{1}{C_{OX}}\right)\right]^{-1}$
Interface trap density	$D_{IT} = \frac{(1\times10^{-12})C_{IT}}{A}$
Mobile ion charge concentration TVS method	$\sum_{-V_{GS}}^{+V_{GS}}(C_{MEAS} - C_{OX})\Delta V_{GS} = Q_O$
STVS method	$N_M = \frac{\sum_{-V_{GS}}^{+V_{GS}}(C_Q - C_H)\Delta V_{GS}}{q}$
Oxide thickness / gate area	$t_{OX} = \frac{A\,\varepsilon_{OX}}{(1\times10^{-19})C_{OX}}$
Series resistance compensation	$C_C = \frac{(G_M^2 + \omega^2 C_M^2)C_M}{a^2 + \omega^2 C_M^2}$ $G_C = \frac{(G_M^2 + \omega^2 C_M^2)a}{a^2 + \omega^2 C_M^2}$ $a = G_M - (G_M^2 + \omega^2 C_M^2)R_{SERIES}$
Threshold voltage	$V_{THRESHOLD} = \left[\pm\frac{A}{10^{12}C_{OX}}\sqrt{4\,\varepsilon_S\, q\lvert N_{BULK}\rvert\lvert\phi_B\rvert} + 2\lvert\phi_B\rvert\right] + V_{FB}$
Work function	$W_{MS} = W_M - \left[W_S + \frac{E_G}{2} - \phi_B\right]$

Table E-15 (continued)
Analysis equations

Analysis function	Equation
Zerbst Plot (Generation Lifetime and Velocity)	$G / n_I = -\varepsilon_S A N_{AVG} C_{OX} \bullet \left[\frac{\frac{1}{C^2_{t(i+1)}} - \frac{1}{C^2_{t(i-1)}}}{n_I t_{\text{int}}} \right] \bullet \left(\frac{1\times10^{12}}{2} \right)$ $w - w_F = 1\times10^{12}\, \varepsilon_S A \left(\frac{1}{C_{ti}} - \frac{1}{C_{OX}} \right) - w_F$

References and bibliography of C-V measurements

References

The references below are cited in this chapter:

Nicollian, E.H. and Brews, J.R., MOS Physics and Technology.

Wiley, New York (1982).

Sze, S.M., Physics of Semiconductor Devices 2nd edition.
Wiley, New York (1985).

Snow, E.H. Grove, A.S., Deal, B.E., and Sah, C.T.J., Ionic Transport Phenomena in Insulating Films, Appl. Phys., 36,
1664 (1965).

Bibliography of C-V Measurements

Texts

Grove, A.S., Physics and Technology of Semiconductor Devices,
Wiley, New York (1967).

Sze, S.M., Semiconductor Devices, Physics and Technology,
Wiley, New York (1985).

Articles and Papers

Feedback Charge Method

Mego, T.J., Improved Feedback Charge Method for Quasistatic CV Measurements in Semiconductors, Rev. Sci. Instr. 57, 11 (1986).

Mego, T.J., Improved Quasistatic CV Measurement Method for MOS, Solid State Technology, 29, 11, 519-21 (1986).

Markgraf, W., Baumann, M., Beyer, A., Arst, P., Rennau, M., Nutzung der statischen CU-Methode im Ranen eines mikrorechnergesteuerten MOS-Messplatzes, Phys. d. Halbleiteroberflaeche, 15, 73 (1984).

Q-V Static Method

Ziegler, K. and Klausmann, E., Static Technique for Precise Measurements of Surface Potential and Interface State Density in MOS Structures, Appl. Phys. Lett. 26, 400 (1975).

Kirov, K., Alexsandrova, S., and Minchev, C., Error in Surface State Determination Caused by Numerical Differentiation of Q-V Data, Solid State Electronics, 18, 341 (1978).

Q-C Method and Simultaneous High-low Frequency C-V

Nicollian, E.H. and Brews, J.R., Instrumentation and Analog Implementation of the Q-C Method for MOS Measurements, Solid State Electronics, 27, 953 (1984).

Boulin, D.M., Brews, J.R., and Nicollian, E.H., Digital implementation of the Q-C Method for MOS Measurements, Solid State Electronics, 27, 977 (1984).

Derbenwick, G.F., Automated C-V and |Y|-w Curves for MOS Device Analysis, Sandia Report SAND80-1308 (1982).

Lubzens, D., Kolodny, A., and Shacham-Diamond, Y.J., Automated Measurement and Analysis of MIS Interfaces in Narrow-Bandgap Semiconductors, IEEE transactions on Electron Devices, ED-28, 5 (1981).

Ramp Method

Kuhn, M., A Quasistatic Technique for MOS C-V and Surface State Measurements, Solid State Electronics, 13, 873 (1970).

Castagne, R., Détermination de la densité d'états lents d'une capacité métak-isolant semiconducteur par l'étude de la charge sour une tension croissant line áirement, C.R. Acad. Sci 267, 866 (1968).

Kerr, D.R., MIS Measurement Technique Utilizating Slow Voltage Ramps, Int. Conf. Properties and Use of MIS Structures, Grenoble, France, 303 (1969).

Castagne, R., and Vapaille, A., Description of the SiO_2-Si Interface Properties by Means of Very Low Frequency MOS Capacitance Measurements, Surface Science, 28, 157 (1971).

Kuhn, M. and Nicollian, E.H., Nonequilibrium Effects in Quasi-static MOS Measurements, J. Electrochem. Soc., 118, 373 (1971).

Lopez, A.D., Using the Quasistatic Method for MOS Measurements, Rev. Sci. Instr. 44, 200 (1973).

Interface States / Doping Profiles

Berglund, C.N., Surface States at Steam Grown Silicon-Silicon Dioxide Interfaces, IEEE Trans. Electron. Dev., 13, 701 (1966).

DeClerck, G., Characterization of Surface States at the Si-SO2 Interface, Nondestructive Evaluation of Semiconductor Materials and Devices (J.N. Zemel, ed.), Plenum Press, New York, p. 105 (1979).

Brews, J.R., Correcting Interface-State Errors in MOS Doping Profile Determinations, J. Appl. Phys. 44, 3228 (1973).

Gordon, B.J., On-Line Capacitance-Voltage Doping Profile Measurement of Low-Dose Ion Implants, IEEE Trans. Dev., ED-27, 12 (1980).

VanGelder, W., and Nicollian, E.H., Silicon Impurity Distribution as Revealed by Pulsed MOS C-V Measurements, J. Electrochem, Soc. Solid State Science, 118, 1 (1971).

MOS Process Characterization

Zaihinger, K.H. and Heiman, F.P., The C-V Technique as an Analytical Tool, Solid State Technology, 13:5-6 (1970).

McMillian, L., MOS C-V Techniques for IC Process Control, Solid State Technology, 15, 47 (1972).

Zerbst, M., Relaxationseffekte an Halbeiter Isolator-Grenzflaechen, Z.Angnew, Phys. 22, 30 (1966).

Mobile Ion Charge Monitoring

Stauffer, L., et al., Mobile Ion Monitoring by Simultaneous Triangular Voltage Sweep, Solid State Technology, 38, S3 (1995).

Appendix F

Using an Agilent 8110A/81110A Pulse Generator

In this section:

Key concepts

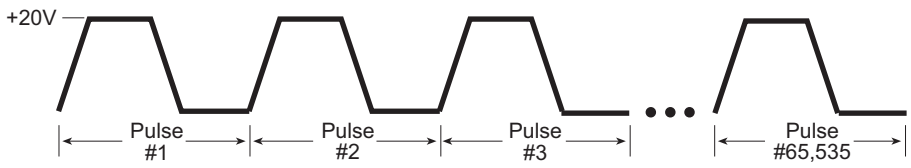
NOTE: *For details on all aspects of the HP pulse generator operation, refer to the HP Model 8110A/81110A User's Manual.*

Pulse generator overview

The Model 4200-SCS can control an HP Model 8110A/81110A Pulse Generator to output from 1 to 65,535 pulses. Figure F-1 shows an example pulse output. Timing parameters that can be set for the output pulse include pulse delay time, pulse width, pulse period, pulse rise time, and pulse fall time. Details on all parameters for the output pulse are provided in hp8110ulib user library reference later in this appendix.

One of the applications for a pulse generator in a semiconductor characterization test system is stress testing. The stress is a burst of pulses applied by the pulse generator to a semiconductor device, such as a flash memory cell. The Model 4200-SCS performs before-stress and after-stress characterization tests on the device.

Figure F-1
Pulse generator output example



Pulse generator tests

The Model 4200-SCS provides the following user modules to perform tests using an HP pulse generator:

- **PguInit8110: Initialization**: Disables the pulse generator output and returns it to a default setup configuration.
- **PguSetup8110: Set up pulse**: Used to define the output pulse.
- **PguTrigger8110: Trigger output**: Used to specify the number of pulses and trigger the pulse output process.

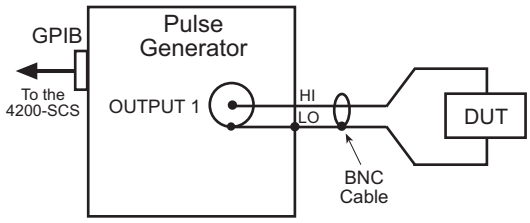
Details on the user modules for the HP pulse generator library are provided in Figures 3-2 later in this appendix.

Connections

Signal connections

Basic signal connections for an output of the pulse generator is shown in Figure F-2. The output LO is connected to the chassis of the pulse generator.

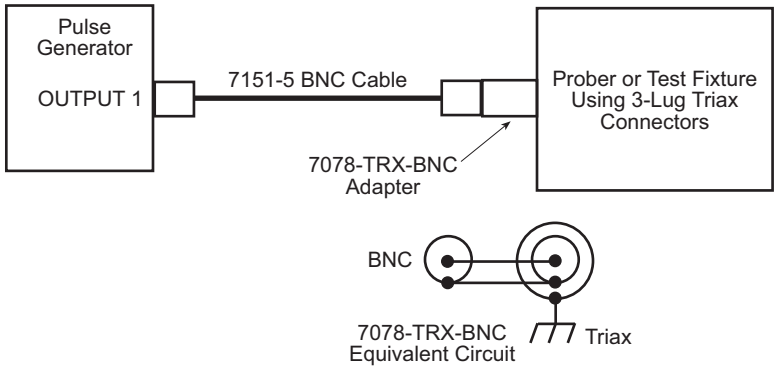
Figure F-2
Basic pulse generator connections to DUT



Triax connections:
Adapters are required to connect the pulse generator to equipment that uses triax connectors (for example, probe station, test fixture, matrix card).

Probe station and test fixture connections:
Figure F-3 shows connections to a probe station or a test fixture that is equipped with 3-slot triax connectors. The Model 7087-TRX-BNC is a 3-lug triax to BNC adapter. As shown, connect the adapter to the 3-slot triax connector and then use a Model 7051-5 BNC cable to make the connection to the pulse generator. Figure F-3 also shows the equivalent circuit for the adapter.

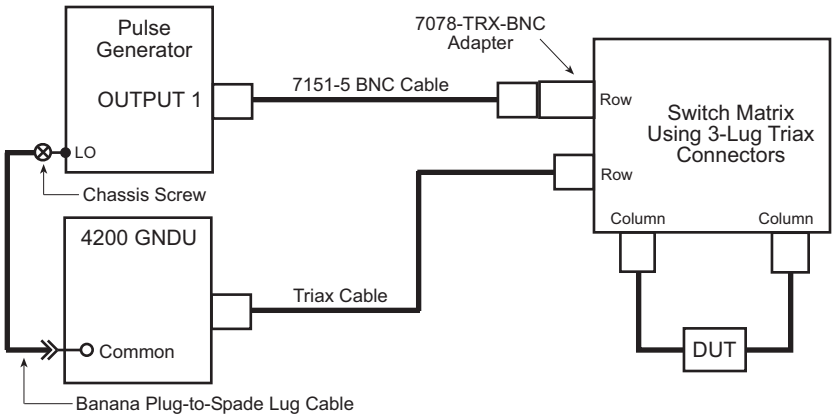
Figure F-3
Connections to prober or test fixture equipped with triax connectors



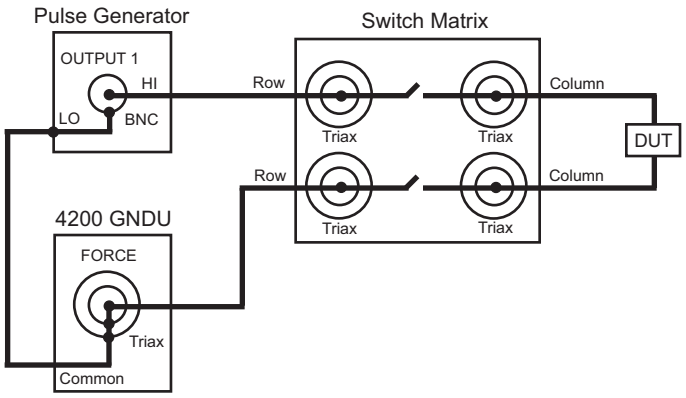
Switch matrix connections:
When using a switch matrix that is equipped with triax connectors, separate HI-to-LO matrix paths are required for the pulse generator. A typical connection scheme for this type of switch matrix is shown in Figure F-4A. As shown, OUTPUT 1(HI) is connected to a matrix row, and the return path (LO) from the switch matrix is connected to the Model 4200 GNDU (ground unit). Note that in order to complete the return path, a separate cable connection from the GNDU to the chassis of the pulse generator is required. Remember, the chassis of the pulse generator is output LO.

Figure F-4B shows the actual pulse output signal path through the switch matrix to the device under test (DUT), and back to the pulse generator. A more detailed look at signal paths is provided in Using Switch Matrices in Appendix B.

Figure F-4
Connections to switch matrix equipped with triax connectors



A) Connections to switch matrix



B) Pulse output signal path

GPIB connections

The Model 4200-SCS controls the pulse generator via the General Purpose Interface Bus (GPIB). Use the Model 7007-1 or 7007-2 GPIB cable to connect the GPIB port of the pulse generator to the GPIB port of the Model 4200-SCS.

Using KCON to add an HP pulse generator to the system

In order for the Model 4200-SCS to control an external instrument, that instrument must be added to the system configuration. The pulse generator is added to the test system using KCON (Keithley CONfiguration utility) as follows:

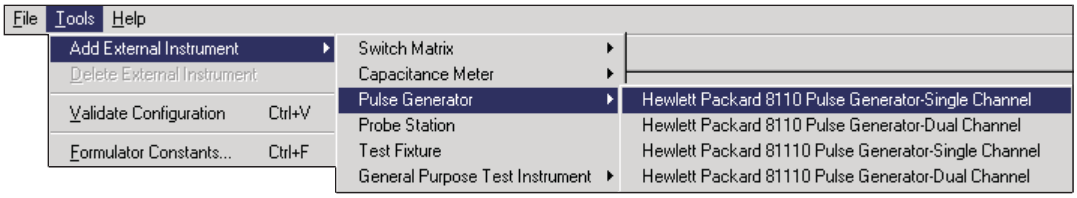
Step 1. Close KITE and open KCON

Close KITE by clicking the close button (X) at the top right-hand corner of the KITE panel. If you have made changes, you will be prompted to save them. On the windows desktop, double-click the **KCON** icon to open KCON. For details on using KCON, see Keithley CONfiguration Utility (KCON) in Section 7.

Step 2. Add pulse generator

Add an HP pulse generator from the **Tools** menu on the Toolbar of the KCON window (Figure F-5). Figure F-6 shows the **Properties & Connections** window for the Model 8110A Pulse Generator-Single Channel.

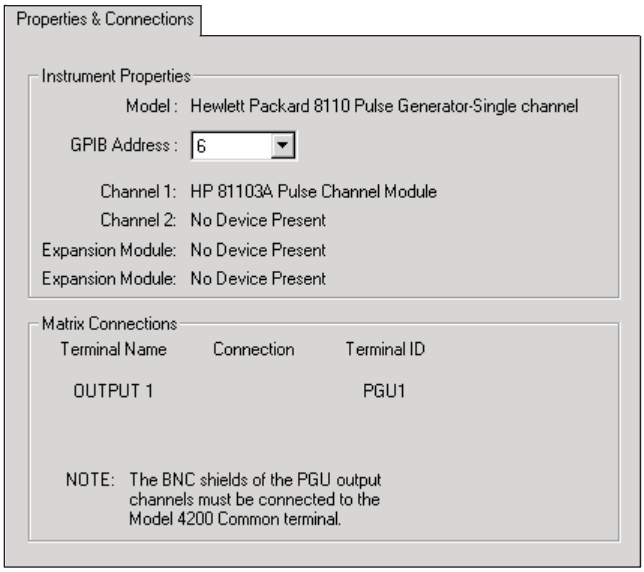
Figure F-5
KCON tools menu to add pulse generator



Step 3. Set GPIB address

The GPIB address setting in the **Instrument Properties** area of the window (Figure F-6) must match the actual GPIB address of the pulse generator.

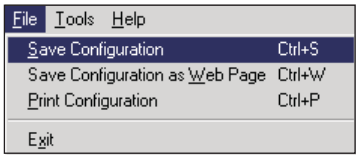
Figure F-6
8110A Pulse Generator single channel Properties & Connections window



Step 4. Save configuration

The KCON configuration is saved from the **File** menu on the toolbar. As shown in Figure F-7, click **Save Configuration**.

Figure F-7
Save KCON configuration



Step 5. Close KCON and open KITE

KCON can be closed from the **File** menu by clicking **Exit**. It can also be closed by clicking the close button (X) at the top right corner of the KCON window.

On the Windows desktop, double-click the **KITE** icon to open KITE.

Pulse generator test example

Stress testing

The pulse generator is used to stress a semiconductor device by applying a burst of pulses to it. The Model 4200-SCS analyzes the effects of the stress by performing before-stress and after-stress characterization tests on the device.

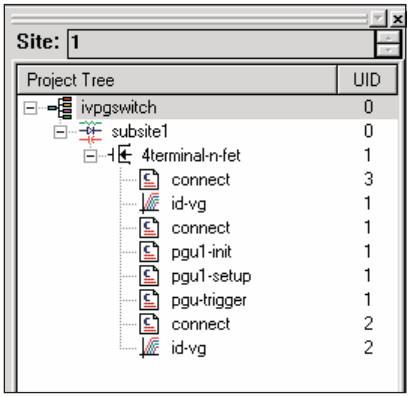
User test modules (UTMs) that utilize the three user modules for the pulse generator (**pgu1-init, pgu1-setup,** and **pgu-trigger**) are located in the **ivpgswitch** project (Figure F-8). The **id-vg** Interactive test module (ITM) is used to graph the transfer characteristics of the DUT.

This project also uses **connect** UTMs to control a switch matrix. If you are not using a switch matrix, the test process can be controlled by manually making connections, and then executing an individual ITM or UTM.

The stress testing process is summarized below:

1. Execute the **id-vd** ITM on the DUT to generate the before-stress characterization graph.
2. Execute the **pgu1-init** UTM to initialize the pulse generator. This UTM uses the **PguInit8110** user module.
3. Use the **pgu1-setup** UTM to define the output pulse. This UTM uses the **PguSetup8110** user module. If using the default parameters for the UTM, the pulse will be configured as shown in Figure F-11.
4. With the pulse generator connected to the DUT, execute the **pgu-trigger** UTM to stress the device. This UTM uses the **PguTrigger8110** user module. The specified number of pulses will be applied to the DUT.
5. Execute the second **id-vd** ITM on the DUT to generate the after-stress graph.
6. Compare the two graphs to determine the effect of the stress.

Figure F-8
ivpgswitch project navigator



hp8110ulib user library reference

The user modules in the **hp8110ulib** user library are used to control an HP pulse generator. These user modules are summarized in Table F-1. Also listed in the table are the Keithley Instruments-created UTM names that use the user modules. Details for each of the user modules follow the table.

Table F-1
hp8110ulib user library

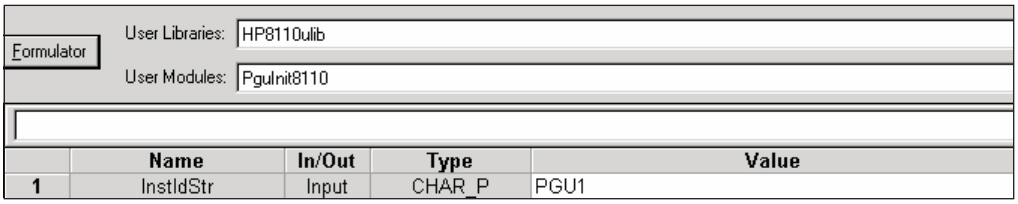
User Module	UTM Name	Description
PguInit8110	pgu1-init	Initializes the pulse generator to the default setup
PguSetup8110	pgu1-setup	Sets the output pulse parameters
PguTrigger8110	pgu-trigger	Specifies pulse count and trigger start of output

PguInit8110 user module

Overview

This user module initializes the pulse generator to a default setup, and is explained in User module description, below. The user module, which is used by the **pgu1-init** UTM, is shown in Figure F-9.

Figure F-9
PguInit8110 (pgu1-init UTM)



User module description

The **PguInit8110** user module initializes the Agilent (HP) 8110A/81110A pulse generator to the following state:

- Disable the output of the specified channel.
- Reset (*RST) to ensure that all errors are cleared.
- Set the output polarity to NORMAL.
- Set the trigger count to 1.
- Set the trigger source to MANUAL.
- Enable SINGLE PULSE mode.
- Allow the rise/fall to be independently programmable.
- Set the pulse height to 0.2 V and base to 0 V.
- Set the rise/fall to 100 e-9 seconds.
- Set the width to 300 e-9 seconds.
- Disable error checking.

Syntax:

status = PguInit8110(char *InstIdStr);

INPUTS:

InstIdStr(char *) The PGU (pulse generator) instrument ID. Valid values for this parameter are PGUx, where x is a number from 1 through 8 (configuration dependent). The PGU instrument ID effectively corresponds to a single pulse generator channel.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0	OK.
-10000(INVAL_INST_ID)	The specified instrument ID does not exist.
-10040(HP8110_NOT_IN_KCON)	No PGU was found in your system's configuration.
-10041(HP8110_NOT_INITED)	The PGU was never initialized.
-10042(HP8110_PULSE_ERROR)	There was an error during pulsing.
-10090(GPIB_ERROR_OCCURRED)	A GPIB communications error occurred.
-10091(GPIB_TIMEOUT)	A time-out occurred during communications.
-10100(INVAL_PARAM)	An invalid parameter was specified.

PguSetup8110 user module

Overview

This user module is used to define the output pulse of the pulse generator. The magnitude and pulse timing parameters that can be set by the user are listed in Table F-2.

Figure F-10 shows the default parameters for the **pgu1-setup** UTM, which uses the **PguSetup8110** user module. Figure F-11 shows the output pulse for that UTM setup.

Details on the user-entered parameters are explained in User module description, later in this appendix.

Table F-2
Pulse magnitude and timing parameters

Parameter	Description [1]	Setting range
BaseValue	The bias (base) voltage level for the pulse	-20 V to +20 V
Amplitude	Amplitude of the pulse, referenced to the BaseValue	-20 V to +20 V
DelayTime	Delay invoked after receiving trigger to start [2]	0 to 0.999 sec
Width	Width of the pulse	3.3nsec to 0.999 sec
Period	Period of the pulse	6.65 nsec to 999 sec [3]
RiseTime	Rise time of the pulse	2 nsec to 0.2 sec
FallTime	Fall time of the pulse	2 nsec to 0.2 sec

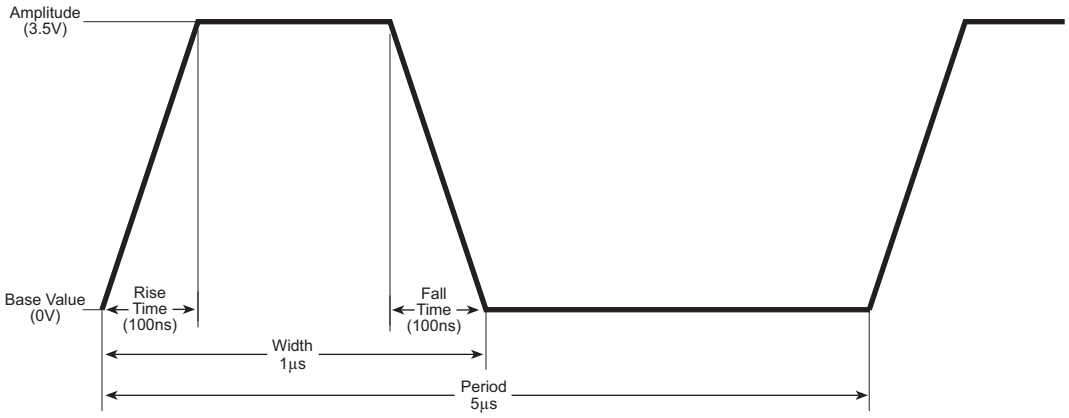
1. Refer to Figure F-11 to complement the description.
2. A DelayTime is not used in the example shown in Figure F-11.
3. Maximum period is 0.999 sec if there is no PLL option installed in the pulse generator.

Figure F-10
PguSetup8110 (pgu1-setup UTM)

Formulator
User Libraries: HP8110ulib
User Modules: PguSetup8110

	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	PGU1
2	DelayTime	Input	DOUBLE	0
3	RiseTime	Input	DOUBLE	100e-09
4	FallTime	Input	DOUBLE	100e-09
5	Width	Input	DOUBLE	1.0e-06
6	Period	Input	DOUBLE	5e-06
7	BaseValue	Input	DOUBLE	0.0
8	Amplitude	Input	DOUBLE	3.500000e+000
9	OutImpedance	Input	INT	0
10	LoadImpedance	Input	DOUBLE	50
11	OutpEnable	Input	INT	1

Figure F-11
pgu1-setup UTM pulse specifications



User module description

The **PguSetup8110** user module defines the pulse timing and voltage settings. Once defined, the pulse can be triggered using the **PguTrigger8110** user module.

Syntax:

status =PguSetup8110(char *InstIdStr, double DelayTime, double RiseTime, double FallTime, double Width, double Period, double BaseValue, double Amplitude, double OutImpedance, double LoadImpedance, double OutpEnable);

INPUTS:

InstIdStr(char *)	The PGU (pulse generator) instrument ID. Valid values for this parameter are PGUx, where x is a number from 1 through 8 (configuration dependent). The PGU instrument ID effectively corresponds to a single pulse generator channel.
DelayTime(double)	The amount of time to delay after receiving the trigger. Valid inputs are from 0 to 0.999 seconds.
RiseTime(double)	Sets the pulse rise time. Valid inputs are from 2 e-09 to 0.2 seconds.
FallTime(double)	Sets the pulse fall time. Valid inputs are from 2 e-09 to 0.2 seconds.

Width(double)	Sets the pulse width. Valid inputs are from 3.3 e-09 to 0.999 seconds.
Period(double)	Sets the period to use if more than one pulse will be triggered. If a single pulse is output (as opposed to a burst of pulses), this parameter is ignored. Valid values range from 6.65e-09 to 999 (0.999 if there is no PLL option installed in the 811[1]0).
BaseValue(double)	The base value of the pulse. If you want a pulse with no DC offset, then set this parameter equal to 0. The valid range of acceptable values for this parameter is -20 V to +20 V.
Amplitude(double)	The amplitude of the pulse as measured from the base value. The valid range of inputs is -20 V to +20 V.
OutImpedance(int)	Sets the PGU's output impedance. Specify 0 for this parameter to set the output impedance to 50 ohms, or specify 1 to set the output impedance to 1000 ohms.
LoadImpedance(double)	The expected impedance of the load (DUT). Valid inputs are 0 through 999e+03 ohms. If unsure, enter the maximum value of 999e+03 ohms.
OutpEnable(int)	A flag that determines whether to enable or disable the PGU's output relay. Specify 1 to enable the output, 0 to disable the output.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

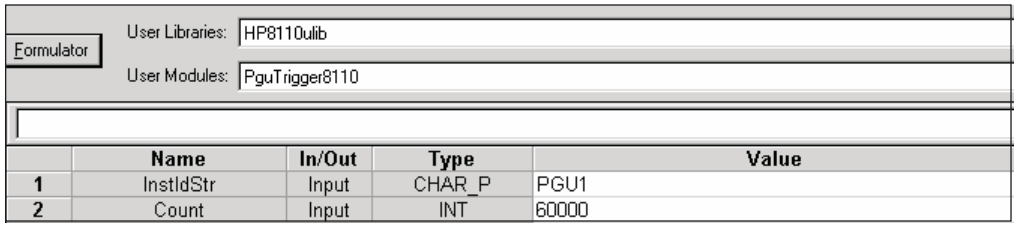
0	OK.
-10000(INVAL_INST_ID)	The specified instrument does not exist.
-10040(HP8110_NOT_IN_KCON)	No PGU was found in your system's configuration.
-10041(HP8110_NOT_INITED)	The PGU was never initialized.
-10042(HP8110_PULSE_ERROR)	There was an error during pulsing.
-10090(GPIB_ERROR_OCCURRED)	A GPIB communications error occurred.
-10091(GPIB_TIMEOUT)	A time-out occurred during communications.
-10100(INVAL_PARAM)	An invalid parameter was specified.

PguTrigger8110 user module

Overview

This user module is used to specify the number of pulses to output, and triggers the start of the pulse output process. The **PguTrigger8110** user module shown in Figure F-12 shows the count (number of pulses) set to 60,000.

Figure F-12
PguTrigger8110



	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	PGU1
2	Count	Input	INT	60000

User-module description

The **PguTrigger8110** function will trigger the pulse (or pulses) previously defined using the **PguSetup8110** function.

Syntax:

status = PguTrigger8110(char *InstIdStr, double Count);

INPUTS:

InstIdStr(char *) The PGU (pulse generator) instrument ID. Valid values for this parameter are PGUx, where x is a number from 1 through 8 (configuration dependent). The PGU instrument ID effectively corresponds to a single pulse generator channel.

Count(double) The number of pulses to output. If Count is > 1, then a burst of pulses with a period as defined in the **PguSetup8110** function will be output. If Count is 1, then a single pulse will be output.

OUTPUTS:

-none-

RETURNED STATUS VALUES:

Returned values are placed in the spreadsheet (**Sheet** tab).

0	OK.
-10000(INVAL_INST_ID)	The specified instrument does not exist.
-10040(HP8110_NOT_IN_KCON)	No PGU was found in your system's configuration.
-10041(HP8110_NOT_INITED)	The PGU was never initialized.
-10042(HP8110_PULSE_ERROR)	There was an error during pulsing.
-10090(GPIB_ERROR_OCCURRED)	A GPIB communications error occurred.
-10091(GPIB_TIMEOUT)	A time-out occurred during communications.
-10100(INVAL_PARAM)	An invalid parameter was specified.

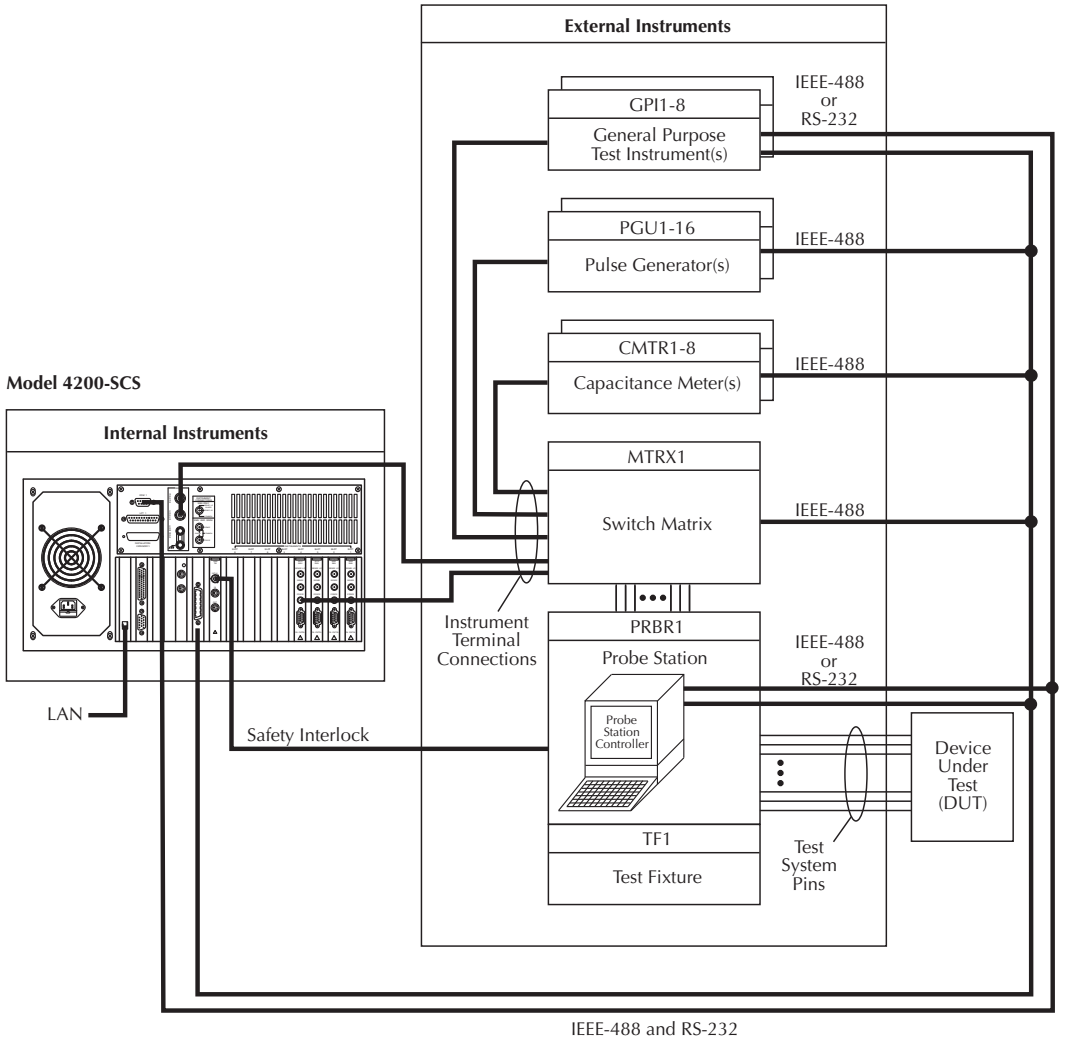
Appendix G
Using a Probe Station

In this section:

Prober control overview

A probe station, like any other external instrument, is controlled by the Keithley Instruments Model 4200-SCS Semiconductor Characterization System® through the use of user modules. Basic system connections are illustrated in Figure G-1.

Figure G-1
Basic system connections



A library of user modules, called **prbgen**, is provided with the Model 4200-SCS to facilitate prober control. This generic prober user library, developed and maintained by Keithley Instruments, allows KITE to control all supported probers in the same manner. Therefore, KITE projects utilizing **prbgen** will work with any prober supported by Keithley Instruments. Refer to Table G-1 for the list of supported probers and where to find additional information.

Table G-1
Supported probers

Supported probe station	Additional information	Appendix
Suss MicroTecModel PA-200	Suss MicroTec PA-200 Prober	H

Table G-1
Supported probers

Supported probe station	Additional information	Appendix
Micromanipulator Model 8860	Micromanipulator 8860 Prober	I
Manual (or Fake)	Appendix J, Using a Manual or Fake Prober	J
Cascade Summit-12000	Cascade Summit-12000 Prober	K
Signatone CM500	Signatone CM500 Prober	L

NOTE: *Contact Keithley Instruments for the most up-to-date list of supported probe stations.*

Sophisticated prober-control software, available from each supported prober vendor, provides access to the full feature set of each prober. In all cases, this prober-control software provides the ability to define a list of wafer locations to be probed. The Model 4200-SCS relies on the prober controller and associated software to maintain this probe list. The **prbgen** user modules communicate with the prober controller via the GPIB bus or COM1 (serial bus) port in most cases, to instruct it to step through the probe list. This technique of prober control is referred to as learn mode because the prober-control software is taught where each probe location is physically located. Table G-2 summarizes the user modules included in the **prbgen** prober control user library.

Table G-2
prbgen user modules

User module	Description
PrInit	Initializes the prober driver and establishes the reference site (or die). All interactive test module (ITM) or user test module (UTM) data acquired by KITE will be tagged with [row, column] site coordinate information that is relative to the reference site.
PrChuck	Instructs the prober to move the probe station into contact or break contact between the wafer and the test system pins (probe needles).
PrSSMovNxt	Instructs the prober to move to the next subsite (or test element group) in the probe list.
PrMovNxt	Instructs the prober to move to the next site (or die) in the probe list.

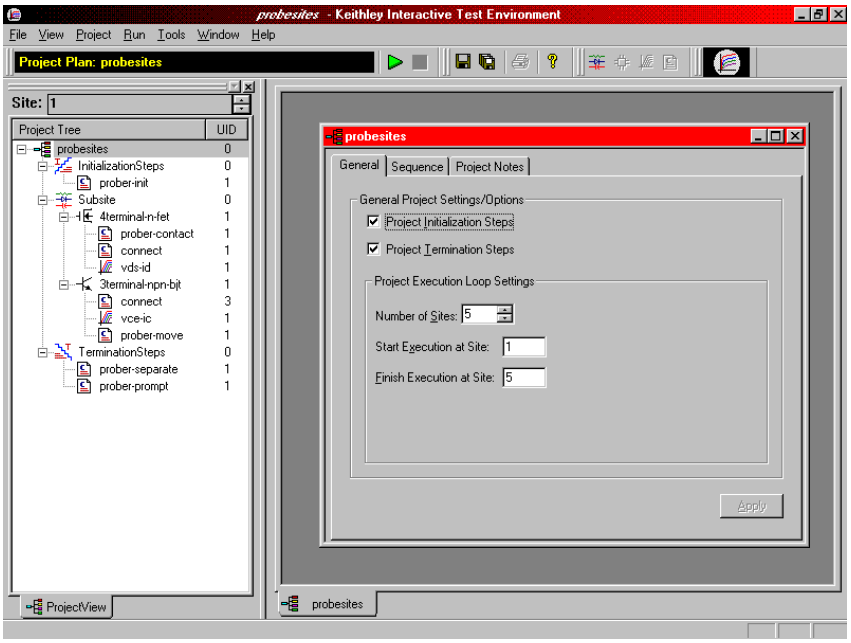
Before a KITE project that utilizes the **prbgen** user library can be executed, the probe list must be created using the appropriate vendor-specific prober-control software. Helpful instructions for creating the probe list for each supported prober are included in this manual (refer to Table G-1).

NOTE: *KCON must have previously been configured to add the prober to the instrument list (refer to the prober-specific appendix listed in Table G-1 for details). To configure a KITE project to control a prober:*

- The KITE project plan must be configured for prober operation (for example, the number of sites to test must be properly defined).
- The KITE project must include the appropriate prober-control UTMs (for example, UTMs connected to **prbgen** user modules).

Figure G-2 shows the project navigator project plan window for the **probesites** KITE project example that follows.

Figure G-2
probesites Project plan



Example test execution sequence: probesites project

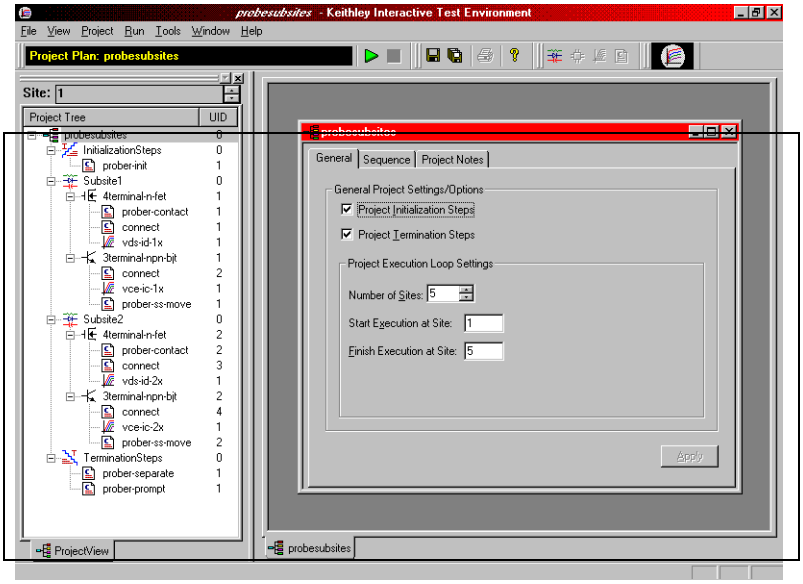
In this example, the pseudo code (Figure G-3) illustrates the test execution sequence that will occur when the **probesites** project is executed from the project level. UTMs that interact with the prober are in bold.

Figure G-3
Example: pseudo code probesites project

```
// Execute the InitializationSteps
EXECUTE prober-init // UTM connected to PrInit
// Execute the Site loop
FOR Site = 1 TO 5
  // Execute the Subsite device tests
     EXECUTE connect
     EXECUTE vds-id-1x
     EXECUTE connect
     EXECUTE vce-ic-1x
     EXECUTE prober-move // UTM connected to PrMovNxt
NEXT Site
// Execute the TerminationSteps
EXECUTE prober-seperate // UTM connected to PrChuck
EXECUTE prober-prompt
```

Figure G-4 shows the project navigator and project plan for the **probesubsites** KITE project example that follows.

Figure G-4
probesubsites Project Plan



Example test execution sequence: probesubsites project

In this example, the pseudo code (Figure G-5) illustrates the test execution sequence that will occur when the **probesubsites** project is executed from the project level. UTMs that interact with the prober are in bold.

Figure G-5
Example: pseudo code probesubsites project

```
// Execute the InitializationSteps
EXECUTE prober-init // UTM connected to PrInit
// Execute the Site loop
FOR Site = 1 TO 5
     // Execute the Subsite1 device tests
     EXECUTE connect
     EXECUTE vds-id-1x
     EXECUTE connect
     EXECUTE vce-ic-1x
     EXECUTE probe-ss-move // UTM connected to PrSSMovNxt
     // Execute the Subsite2 device tests
     EXECUTE connect
     EXECUTE vds-id-2x
     EXECUTE connect
     EXECUTE vce-ic-2x
     EXECUTE probe-ss-move // UTM connected to PrSSMovNxt
NEXT Site
// Execute the TerminationSteps
EXECUTE prober-seperate // UTM connected to PrChuck
EXECUTE prober-prompt
```

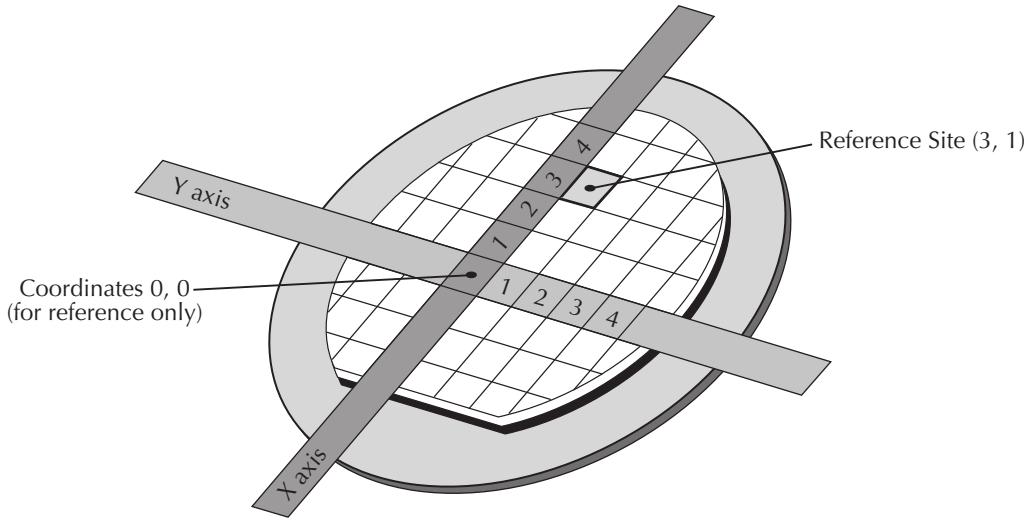
Understanding site coordinate information

Reference site (die)

The designated reference site is defined with `PrInit(_, _, _, columns, rows, _, _)`. This is the prober's first stopping point once aligned. The reference site's physical location may be any coordinate selected on the wafer and is selected for probing or marked for probing through the prober's software. The wafer's coordinate system is also defined through the prober's software. For example, the reference site's coordinates shown in Figure G-6 are (3, 1).

NOTE: *The reference site defined must agree with the physical location of the wafer. This is the location on the wafer directly below the probe pins after the wafer has been loaded.*

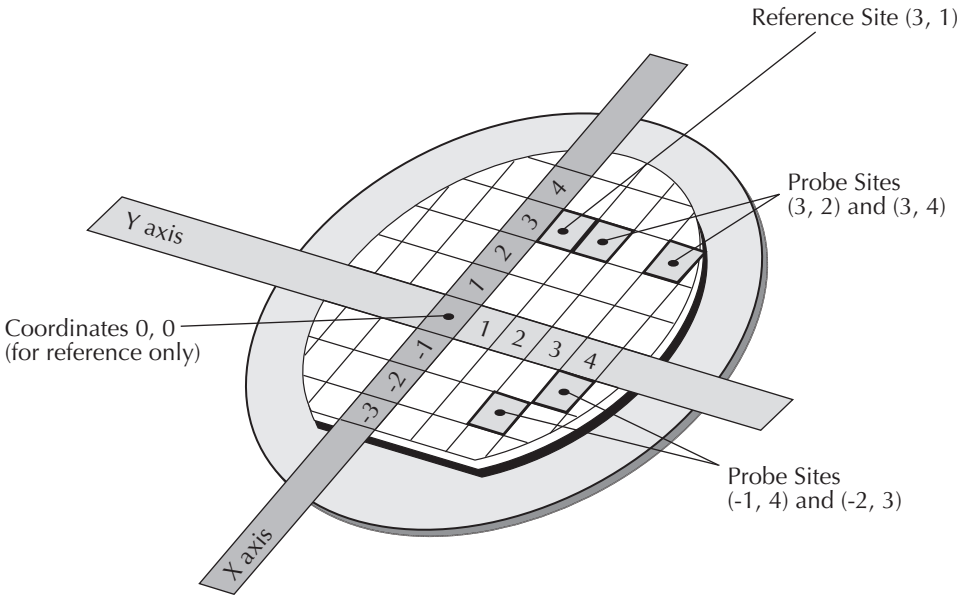
Figure G-6
Sample reference site location



Probe sites (die)

Dies marked as probe sites (in the prober software) define the areas to be tested. The probe sites' physical location can be any coordinates selected on the wafer. Marking a die as a probe site also selects the site for probing. Each probe site's coordinates are referenced with respect to the reference site's coordinates. For example, with the reference site of (3, 1), the coordinates of the five probe sites shown in the Figure G-7 are (3,1), (3, 2), (3, 4), (-1, 4), and (-2,3).

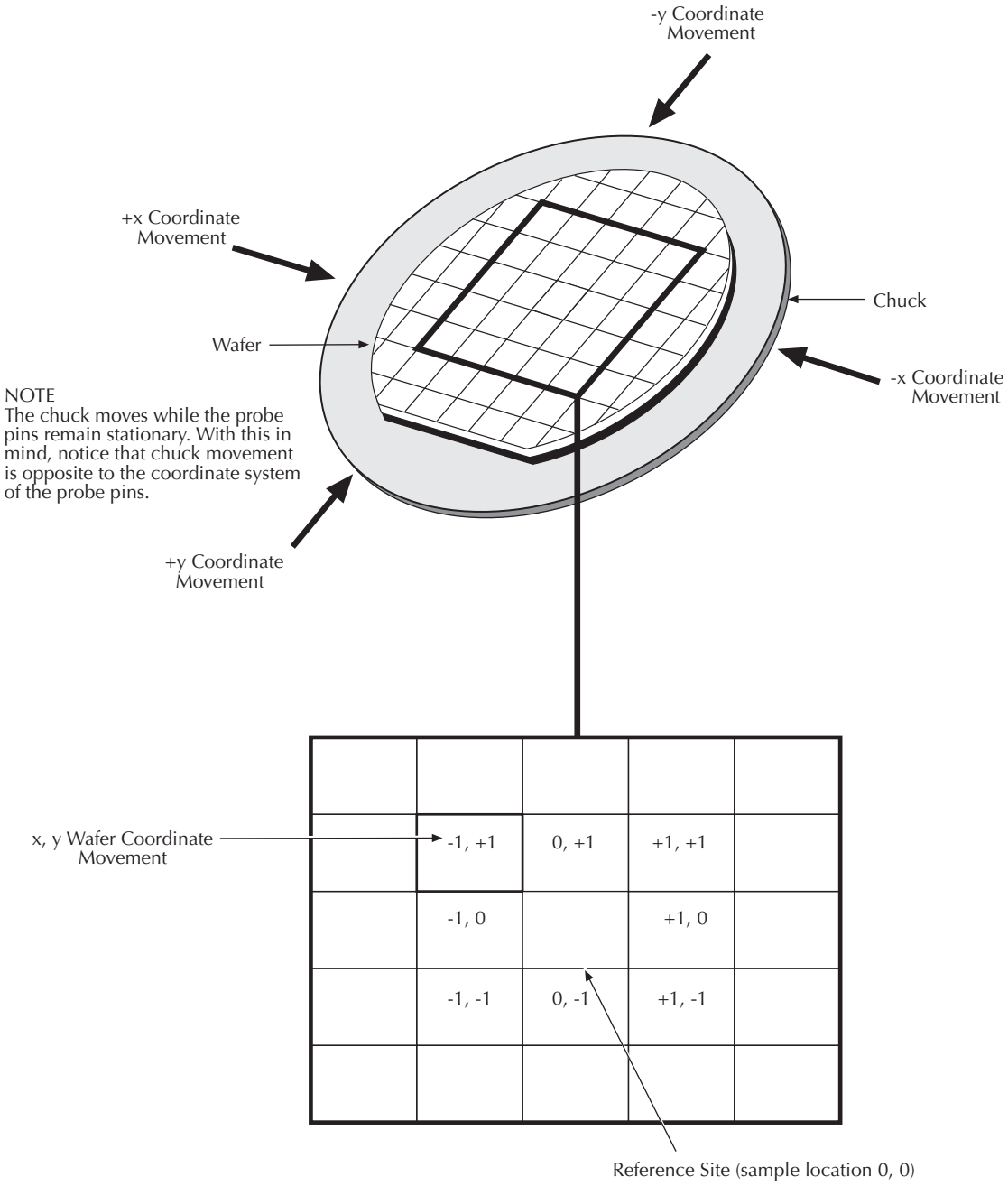
Figure G-7
Sample probe site location



Chuck movement

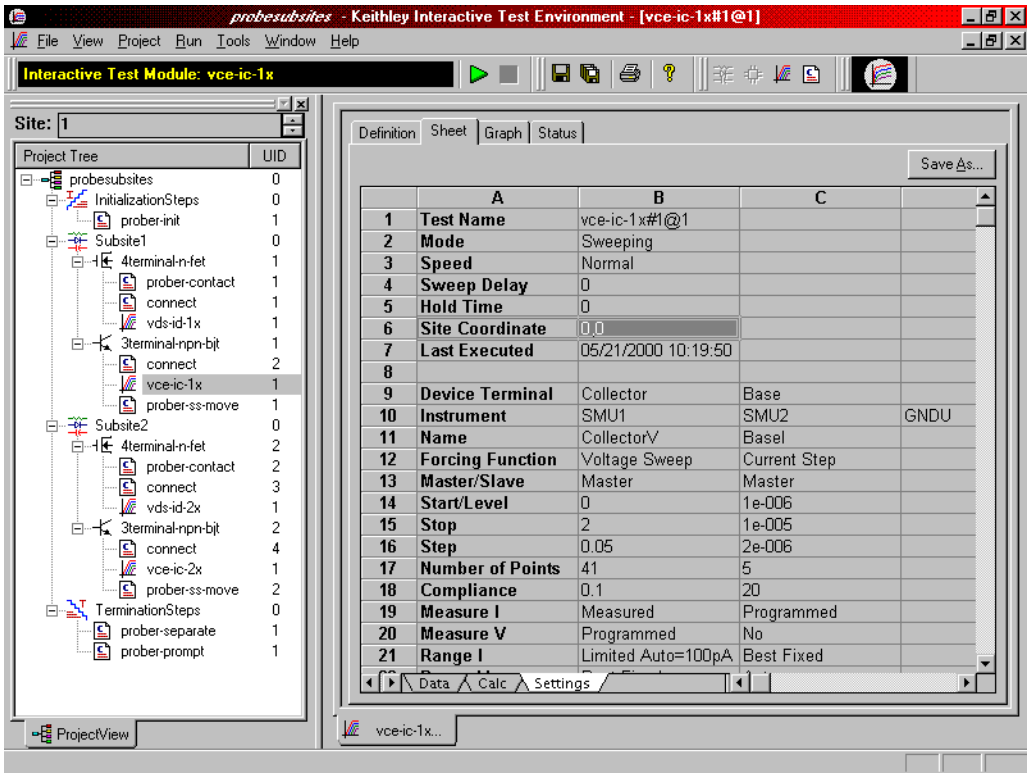
Coordinate movements are described using a first quadrant coordinate system and x, y coordinates (+x values move East and +y values move North). To accommodate this system, the correct quadrant must be configured (prober dependent). Applicable quadrant setup instructions are contained in the applicable appendix (prober specific). When specifying chuck movements, use the coordinates of the desired site. The chuck will automatically move in the proper direction to position the probe pins over the correct die. For example, to move from the reference site to the die up one and over one, command the chuck to move (1, 1). Refer to Figure G-8 for a representation of the relationship between chuck movement and (x, y) coordinates.

Figure G-8
Chuck movement



At the conclusion of a given test, the site coordinates are recorded in the sheet settings. These coordinates will only be valid if a project utilizes remote prober control (real prober). The coordinate system will be based on the 4th and 5th parameters of the **PrInit** call (only in the initialization step). The site coordinates will change only after a site movement is performed; the coordinates will change once the bottom of the project loop is reached. At the top of each iteration, the site coordinates will remain the same until the site movement is performed. Refer to Figure G-9 for an example of a table containing site coordinates (see column B, row 5).

Figure G-9
KITE: Example of site coordinates: Sheet tab



prbgen User Library Reference

This section contains a detailed description of the available commands in the generic prober library.

PrSSMovNxt

Purpose In learn mode, the **PrSSMovNxt** command causes the prober to move to the next SUB-SITE after inking (if desired), using the ink_number.

Format

```
PrSSMovNxt(int ink_number);
```

Where: `ink_number` (not supported for the Model 4200-SCS) = an integer between 0 and 15. Its value will determine which of four inkers will fire. The binary value of the `ink_number` will have four bits. If the lowest order bit is set, inker number 1 will fire. If the second lowest order bit is set, inker number 2 will fire. If the next bit is set, inker number 3 will fire, and if the highest order bit of this number is set, inker number 4 will fire. So, for example, if ink_number is equal to 9, the lowest and the highest order bits are set so inkers 1 and 4 will fire.

Example

```
status = PrSSMovNxt();
```

Remarks LIBRARIES:

Dependencies: PRBCOM

The return value of this function is interpreted in Table G-3, below.

Table G-3
PrSSMovNxt function return values

Symbol	Value
Success:	
PR_OK	1
Errors:	
MOVE_FAIL	-1014
BAD_MODE	-1011
UNEXPE_ERROR	-1015
UNINTEL_RESP	-1013

PrMovNxt

Purpose In learn mode, the **PrMovNxt** command causes the prober to move to the next SITE after inking (if desired), using the ink_number.

Format
```
PrMovNxt(int ink_number);
```

Where: `ink_number` (not supported for the Model 4200-SCS) = an integer between 0 and 15. Its value will determine which of four inkers will fire. The binary value of the `ink_number` will have four bits. If the lowest order bit is set, inker number 1 will fire. If the second lowest order bit is set, inker number 2 will fire. If the next bit is set, inker number 3 will fire, and if the highest order bit of this number is set, inker number 4 will fire. So, for example, if `ink_number` is equal to 9, the lowest and highest order bits are set so inkers 1 and 4 will fire.

Example
```
status = PrMovNxt();
```

Remarks LIBRARIES:

Dependencies: PRBCOM

The return value of this function is interpreted in Table G-4, below.

Table G-4
PrMovNxt function return values

Symbol	Value
Success:	
PR_OK	1
Errors:	
MOVE_FAIL	-1014
BAD_MODE	-1011
UNEXPE_ERROR	-1015
UNINTEL_RESP	-1013

PrInit

Purpose This command initializes the prober with die size, first coordinate (X & Y), units (mm or mils) and mode information.

Format
```
PrInit (int mode, double x_die_size,
```

```
double y_die_size, int x_start_position,
int y_start_position, int units, int sub_type);
```

Where:

`ink_number` (not supported for the Model 4200-SCS) = An integer between 0 and 15. Its value will determine which of four inkers will fire. The binary value of the `ink_number` will have four bits. If the lowest order bit is set, inker number 1 will fire. If the second lowest order bit is set, inker number 2 will fire. If the next bit is set, inker number 3 will fire, and if the highest order bit of this number is set, inker number 4 will fire. So, for example, if ink_number is equal to 9, the lowest and the highest order bits are set so inkers 1 and 4 will fire.

`int mode` = The mode to be used with a specific prober. External mode is used when the tester explicitly directs all of the actions the prober is to perform. Learn mode is used when the prober is configured with ALL of the wafer stepping information. The tester in this case will simply command the prober to perform the next operation. Please confirm the correct mode of operation for each specific application. Supported modes vary from prober to prober.

- 1 Manual (typically manual style probers)
- 2 External (typically auto style probers)
- 6 Learn (typically semi-automatic style probers)

`double x_die_size, double y_die_size` = The second and third parameters of this function are entered as double precision numbers. The units used to express these values depend on the value of the units parameter specified later in this function. If the units are metric, these parameters are input in millimeters. If the units selected are English, these parameters are input in mils.

`int x_start_position, int y_start_position` = (the fourth and fifth parameters of this function) Integer values that are the x and y locations to be used to define the prober position at alignment.

`int units` = The units specification parameter. This parameter is an integer. If this parameter is 0 then the units selected are English. If this parameter is 1 then the units selected are metric.

`int sub_type` = An integer that represents the secondary type of the prober (not supported for the Model 4200-SCS).

Example

```
status = PrInit (,2,2,1,1,1,);
```

Remarks

LIBRARIES:

Dependencies: PRBCOM

The return value of this function is interpreted as follows:

Table G-5
PrInit function return values

Symbol	Value
Success:	
PR_OK	1
Errors:	
UNINTEL_RESP	-1013
UNEXPE_ERROR	-1015
SET_MODE_FAIL	-1006
INVAL_PARAM	-1027

PrChuck

Purpose This command will direct the prober to have the probe pins make CONTACT with the wafer or SEPARATE the pins from the wafer.

Format

```
PrChuck (int chuck_position);
```

Where:The parameter `chuck_position` is an integer (INTEGER 0-1) :

Table G-6
PrChuck function return values

Chuck movement	Chuck position value
Separation:	0
Contact:	1

To cause SEPARATION from the chuck, set chuck_position = 0.
To cause CONTACT with the chuck, set chuck_position = 1.

Example

```
status = PrChuck (int 1);
```

Remarks LIBRARIES:

Dependencies: PRBCOM

The return value of this function is interpreted as follows:

Table G-7
PrChuck function return values

Symbol	Value
Success:	
PR_OK	1
Errors:	
BAD_CHUCK	-1017
UNINTEL_RESP	-1013
INVAL_MODE	-1008

Appendix H
Suss MicroTec PA-200 Prober

In this section:

Required probe station software

The following six programs are used to configure and operate the PA-200 prober with the Keithley Instruments Model 4200-SCS (all are required):

- **ProberBench NT**: Provides easy access to configuration and help programs.
- **Wafer Map**: Use to configure wafer geometry, set origin, set home, select dies to probe, and align the wafer.
- **Chuck Navigator**: Use to move the chuck and select subsite(s).
- **PB-GPIB**: Use to configure the GPIB interface.
- **PB-RS-232**: Use to configure the serial interface.
- **Prober Setup (with in the service programs folder)**: Use to initialize the serial communications port.

Software versions

The following list contains the software versions used to verify the configuration of the PA-200 prober with the Model 4200-SCS:

Product Name: WaferMap for ProberBench NT
Product Version: 3.1 (Feb 12, 1999)
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved
Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Product Name: NI-GPIB Driver for ProberBench NT
Product Version: 3.10 (Jan 21, 1999)
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved
Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Product Name: PBRS232 Interface for ProberBench NT
Product Version: 3.00
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved
Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Product Name: Navigator for ProberBench NT
Product Version: 3.1 (Feb 1, 1999)
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved
Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Product Name: TableView for ProberBench NT
Product Version: 3.1 (Feb 3, 1999)
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved
Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Product Name: Remote Communicator for ProberBench NT
Product Version: 3.00
Copyright: Copyright © Karl Suss 1998 - All Rights Reserved

Kernel: 3.000000 ProberBench Kernel Version 3.10 12-7-98
Control Box: 2.400000 ProberBench Control Box 2.4

Probe station configuration

CAUTION **Although this appendix provides instructions on prober setup and configuration, make sure that you are familiar with the Suss MicroTec PA-200 Prober and its supporting documentation before attempting setup, configuration, or operation.**

There are four general steps required to set up and configure the PA-200 prober for use with the Model 4200-SCS:

Step 1. Set up communication

Step 2. Set up wafer geometry

Step 3. Create a site definition and define a probe list

Step 4. Load, align, and contact the wafer

Each step is detailed later in this section.

Modifying the prober configuration file

NOTE: *This file is modified using the Model 4200-SCS computer.*

The default prober configuration file is contained in Figure H-1. As shown, the file is configured for use with a serial prober setup. To configure the prober for use with a GPIB communication setup, use Notepad.exe to comment out the lines (with #) in the `Configuration for PA200 probers:` section and activate the lines (remove the #) in the `Configuration for direct GPIB probers:`. Ensure that you only activate the lines that start with `PROBER_1_...`.

Configuration file location: `C:\S4200\sys\dat\prbcnfg_PA200.dat`

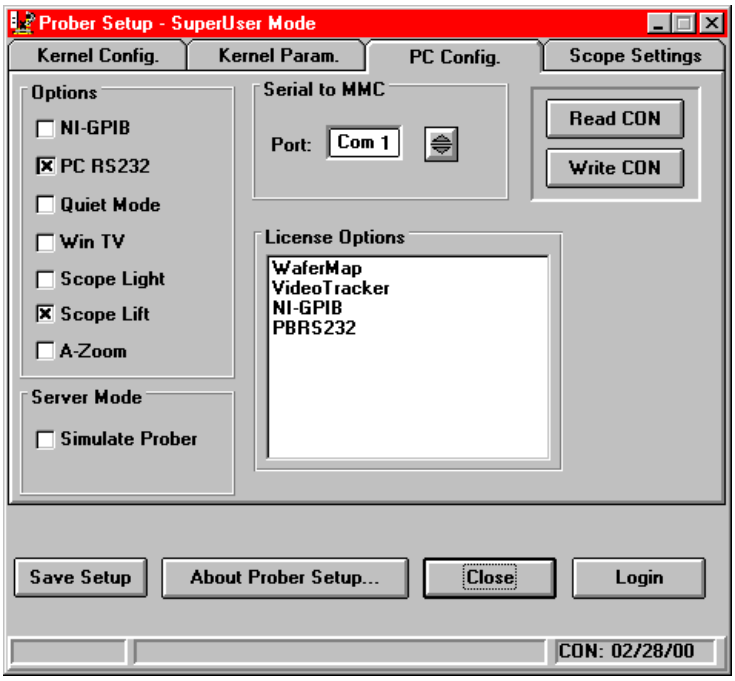
Figure H-1
Sample PA-200 prober configuration file

```
# prbcnfg_PA200.dat  - DEFAULT Prober Configuration File
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#        01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#   OcrPresent
#   AutoAlnPresent
#   ProfilerPresent
#   HotchuckPresent
#   HandlerPresent
#   Probe2PadPresent
#
#
# Configuration for PA200 probers:
#   PA200
#
PROBER_1_PROBTYPE=PA200
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=SERIAL
PROBER_1_DEVICE_NAME=COM1
PROBER_1_BAUDRATE=9600
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
#
#
# Configuration for direct GPIB probers:
#   PA200
#
#PROBER_1_PROBTYPE=PA200
#PROBER_1_OPTIONS=0,0,0,0,1,0
#PROBER_1_IO_MODE=GPIB
#PROBER_1_GPIB_UNIT=0
#PROBER_1_GPIB_SLOT=1
#PROBER_1_GPIB_ADDRESS=8
#PROBER_1_GPIB_WRITEMODE=0
#PROBER_1_GPIB_READMODE=2
#PROBER_1_GPIB_TERMINATOR=13
#PROBER_1_TIMEOUT=300
#PROBER_1_SHORT_TIMEOUT=5
#PROBER_1_MAX_SLOT=25
#PROBER_1_MAX_CASSETTE=1
#
#
```

Step 1. Set up communication

The Suss MicroTec PA-200 prober may be configured for serial or GPIB communication. Ensure that the prober configuration file is set up properly for the type of desired communication interface (see Modifying the prober configuration file earlier in this appendix).

Figure H-2
Prober setup: PC Config tab



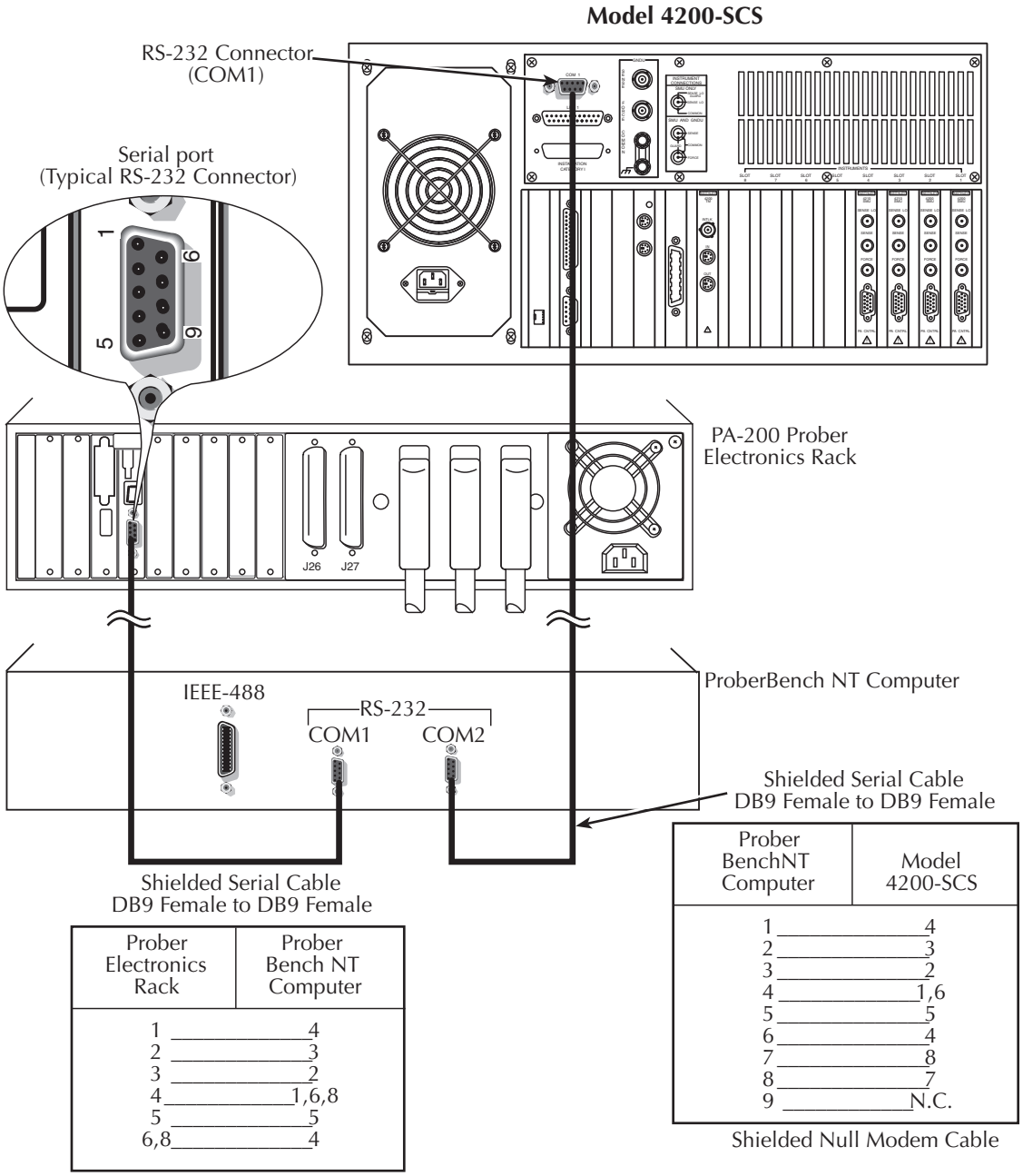
Serial

NOTE: *The following configuration is accomplished using ProberBench NT computer.*

1. Connect the ProberBench NT computer's COM2 port to the Model 4200-SCS COM1 port using a DB9 female to DB9 female cable (shielded null modem cable) (Figure H-3). Also ensure that the ProberBench NT computer serial port (COM1) is connected to the PA-200 Prober Electronics Rack serial port.
2. Double-click the ProberBench NT icon (shortcut) on desktop.
3. Double-click the **Service Programs** file.
4. Double-click the **Prober Setup** file in the **Service Programs** directory.
5. Select the **PC RS232** option.
6. Make sure **Simulate Prober** box in the Server Mode section of the dialog box is **unchecked**.
7. Click the **Save Setup** button.
8. Click **Close**.
9. From the ProberBench NT window, double-click the **PB-RS232** file.

NOTE: *COM2 is used for communications between the 4200 ProberBench NT. COM1 is used for communications between the ProberBench NT and the electronics rack.*

Figure H-3
Model 4200-SCS and PA-200 serial port connection

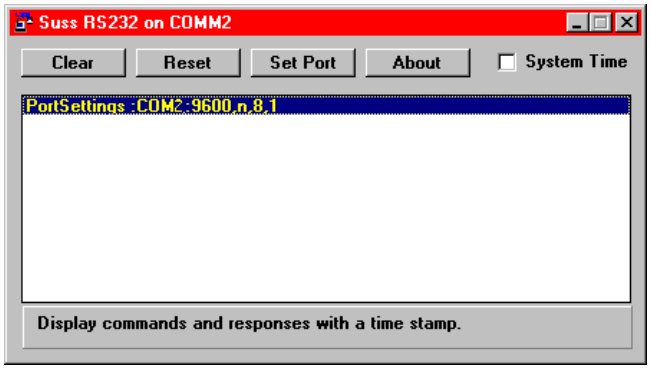


10. From the **Suss RS232 on COMM2** dialog box (Figure H-4), click **Set Port**.
    - Set communication protocol to: 9600, n, 8, 1 for serial port COM2.
    - Make sure **Disable COM Port** is NOT checked.

NOTE: *Leave the **Suss RS232 on COMM2** dialog box open (Figure H-4). This will ensure its services are available for the WaferMap program.*

11. Click **Save/Exit**.
12. From the **Suss RS232 on COMM2** dialog box, click **Reset**.

Figure H-4
Suss RS232 on COMM2 dialog box



GPIB

NOTE: *The following configuration is accomplished using ProberBench NT computer.*

1. Connect the Model 4200-SCS GPIB port and the ProberBench NT computer's GPIB port using a shielded IEE-488 cable (Model 7007). Refer to Figure H-5 and Table H-1 for pinouts, and to Figure H-7 for a connection diagram.

Figure H-5
IEEE-488 connector

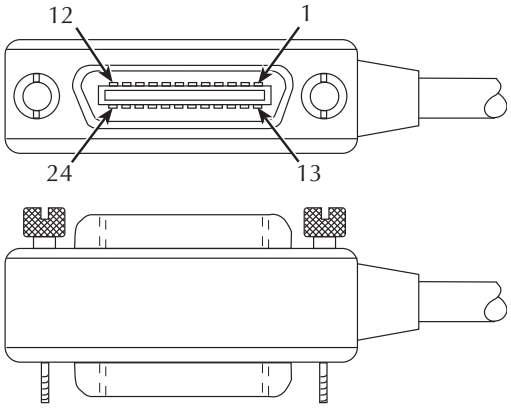


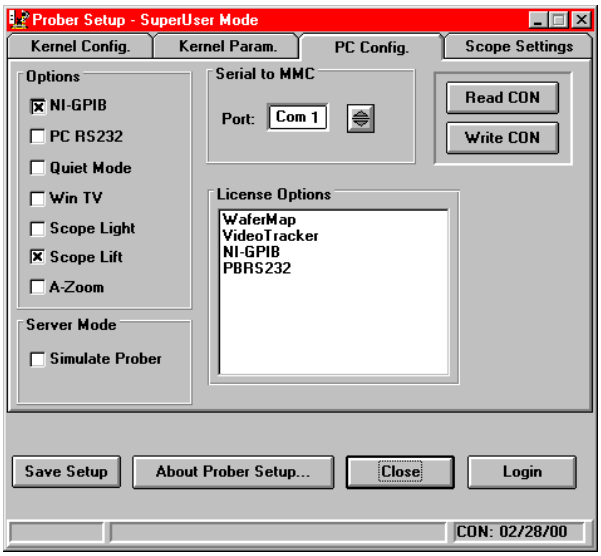
Table H-1
IEEE-488 control connector terminals

Contact number	IEEE-488 designation	Type
1	DI01	Data
2	DI02	Data
3	DI03	Data
4	DI04	Data
5	EOI (24)*	Management
6	DAV	Handshake
7	NRFD	Handshake
8	NDAC	Handshake
9	IFC	Management
10	SRQ	Management
11	ATN	Management
12	SHIELD	Ground
13	DI05	Data
14	DI06	Data
15	DI07	Data
16	DI08	Data
17	REN (24)*	Management
18	Gnd (6) *	Ground
19	Gnd (7) *	Ground
20	Gnd (8) *	Ground
21	Gnd (9) *	Ground
22	Gnd (10) *	Ground
23	Gnd (11) *	Ground
24	Gnd, LOGIC	Ground

* Numbers in parentheses refer to signal ground return of referenced contact number. EOI and REN signal lines return on contact 24.

2. Double-click the ProberBench NT icon (shortcut) on desktop.
3. Double-click the **Service Programs** file.
4. Double-click the **Prober Setup** file in the **Service Programs** directory. The Prober Setup window appears (Figure H-6).
5. Check the **NT-GPIB Option**.

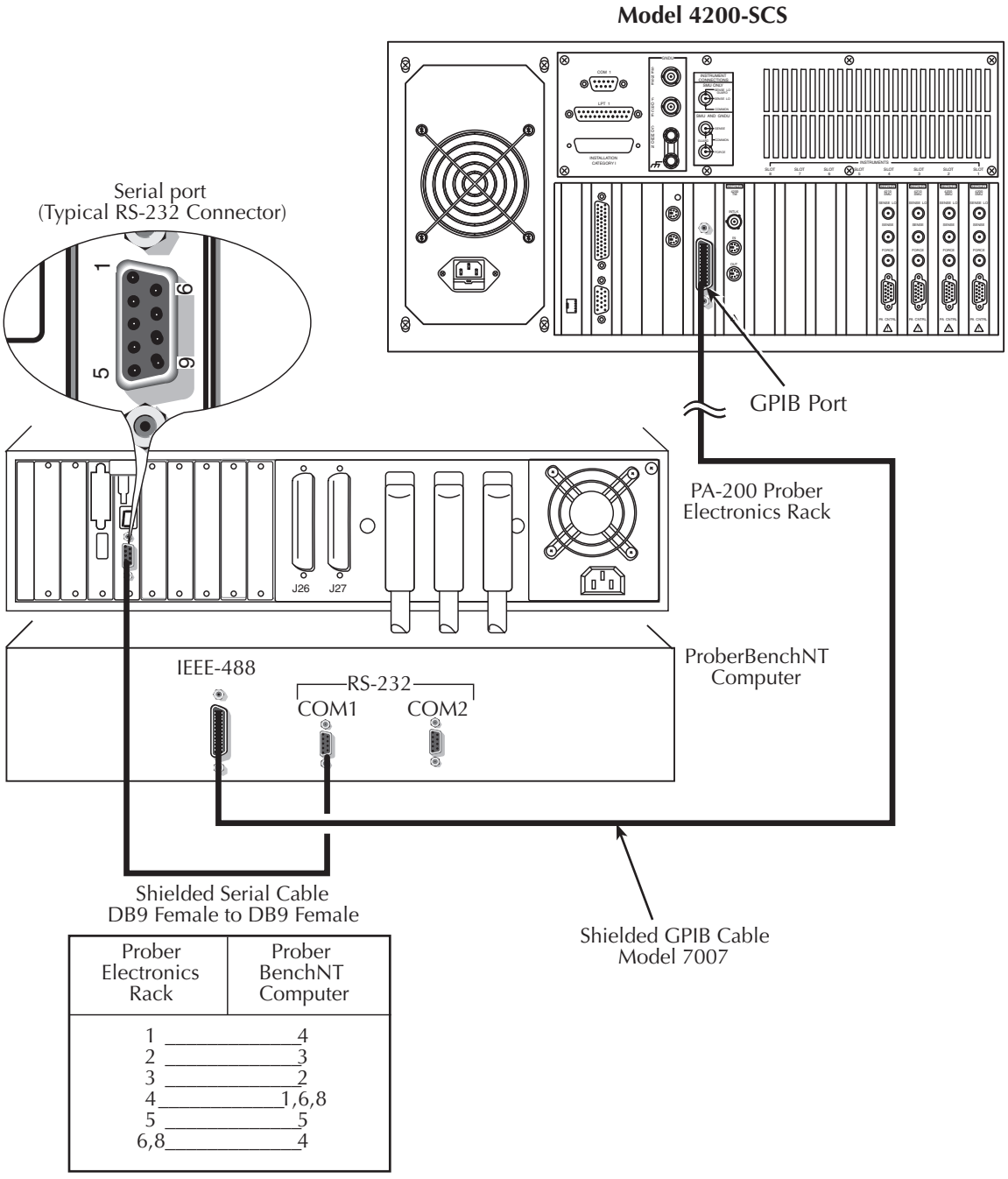
Figure H-6
Prober setup: PC Config tab



6. Ensure that the Simulate Prober box in the Server Mode section of the dialog box is not checked.
7. Click the **Save Setup** button.

NOTE: *Do not use the GPIB port on the Prober Electronics Rack. Make sure to connect the cable between the Model 4200-SCS and the ProberBench NT computer's GPIB ports as shown.*

Figure H-7
Model 4200-SCS and PA-200 IEEE-488 connection



Prober Electronics Rack	Prober BenchNT Computer
1	4
2	3
3	2
4	1,6,8
5	5
6,8	4

8. From the ProberBench NT window, double-click the **PB-GPIB** file (Figure H-8).
9. From the ProberBench GPIB Interface, select **Interface Driver** from the **Configure** pull-down (Figure H-9).

Figure H-8
ProberBench NT window

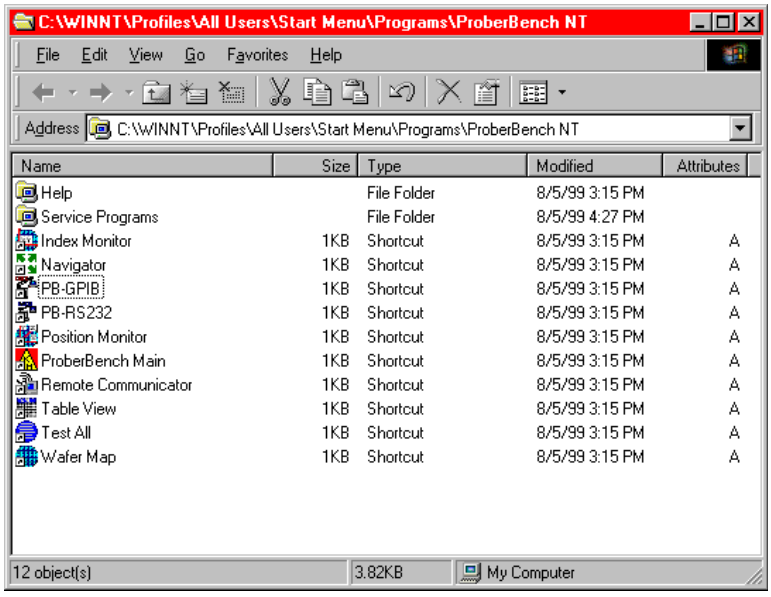
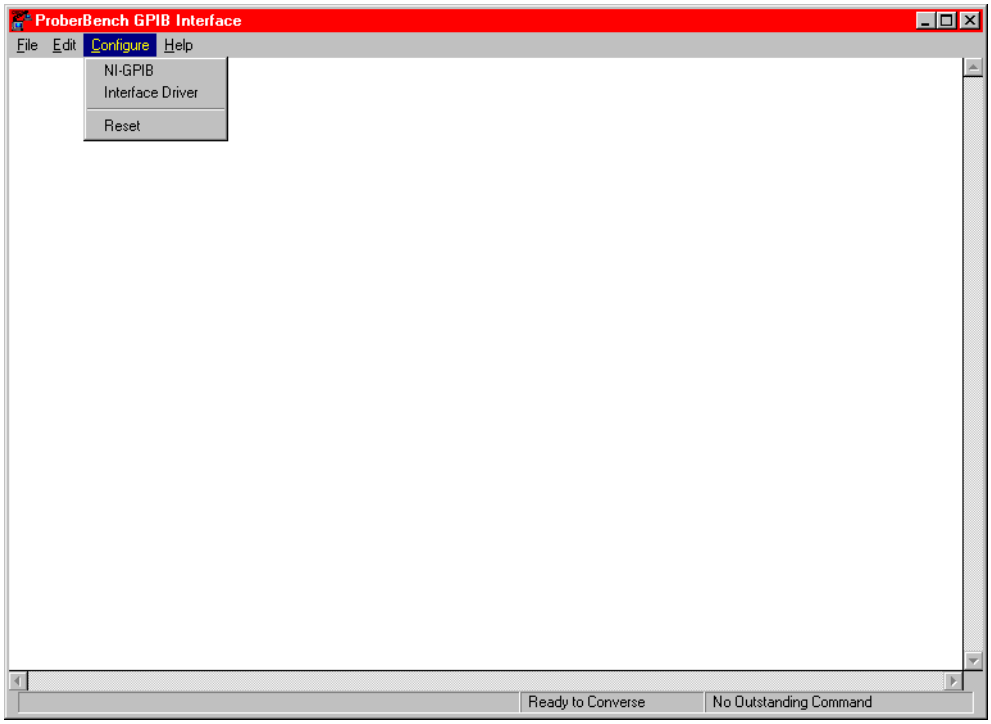
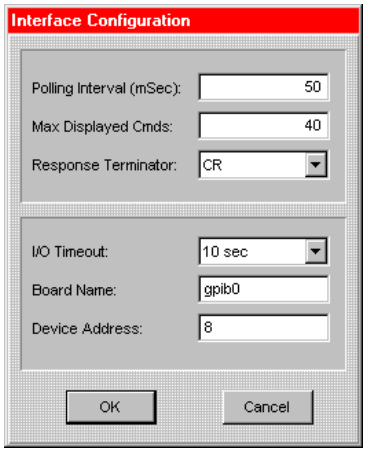


Figure H-9
ProberBench GPIB interface



10. From the **Interface Configuration** window (Figure H-10), change **Response Terminator** to **CR**.
11. GPIB only: Ensure that the GPIB address matches the address contained in the configuration file.

Figure H-10
Interface Configuration window



12. Click **OK**.

NOTE: *Leave the **Suss RS232 on COMM2** dialog box open (Figure H-4). This will ensure its services are available for the WaferMap program.*

Step 2. Set up wafer geometry

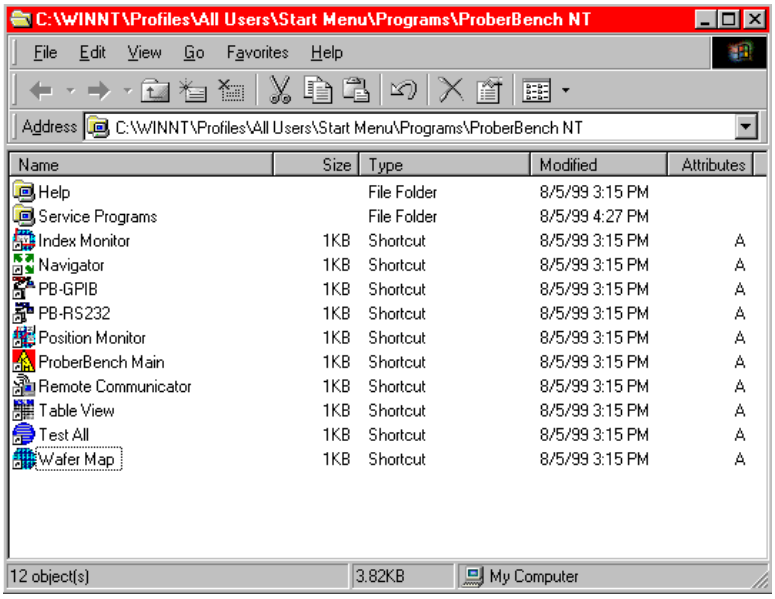
NOTE: *The following configuration is accomplished using ProberBench NT computer.*

1. Select the **ProberBench NT** icon (shortcut) on desktop (Figure H-11).

Figure H-11
ProberBench NT icon

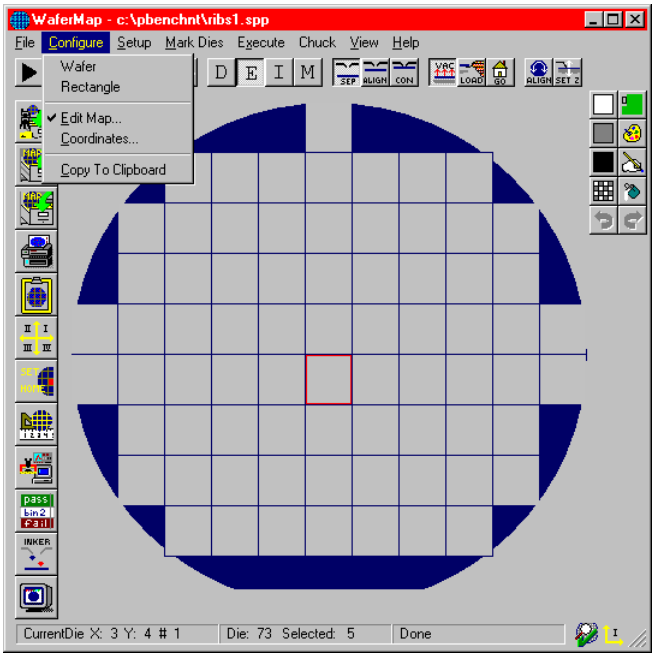
2. From the ProberBench NT window, select **WaferMap** file (Figure H-12).

Figure H-12
ProberBench NT window



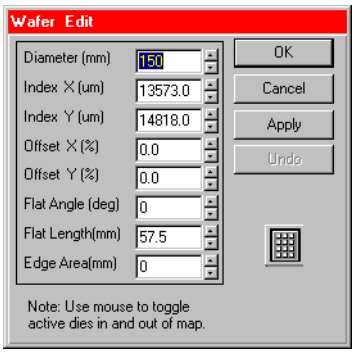
3. From the WaferMap window, create or open a WaferMap (Figure H-13).
4. Select **Edit Map** from the **Configure** pull-down (Figure H-13).

Figure H-13
WaferMap window



5. Enter wafer geometry (Figure H-14).
   - Enter values.
   - Click **Apply**.
   - Click **OK**.

Figure H-14
Wafer Edit dialog

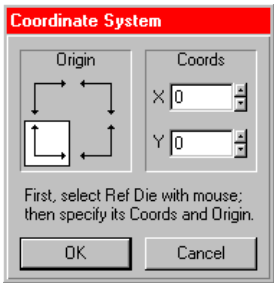


6. Select **Coordinates** from the **Configure** pull-down (Figure H-15).

Figure H-15
Configure pull-down

7. From the **Coordinate System** dialog box, set **Origin** as shown (set any initial **X** and **Y** Coordinates, see Figure H-16).

Figure H-16
Coordinate System dialog box



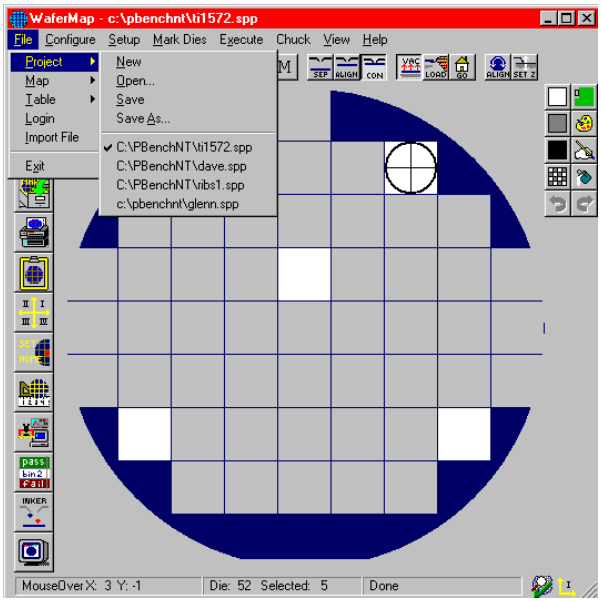
8. Click **OK**.

NOTE: *Refer to the* **probesites** and **probesubsites** KITE project *examples for specifics on selecting sites to probe.*

---

9. **Save** the WaferMap settings (Figure H-17).

Figure H-17
pa200 WaferMap: save



Step 3. Create a site definition and define a probe list

NOTE: *The following setup procedure is accomplished using ProberBench NT computer.*

Creating a site definition for single subsites per die involves using the software to create a selection of dies to probe. If a single subsite per die is to be probed, refer to probesites KITE project example later in this appendix. Creating a site definition for multiple subsites per die also involves using the software to create a selection of dies to probe, but also includes creating a selection of the subsites on each die that will be probed. If multiple subsites per die will be probed, refer to probesubsites KITE project example later in this appendix.

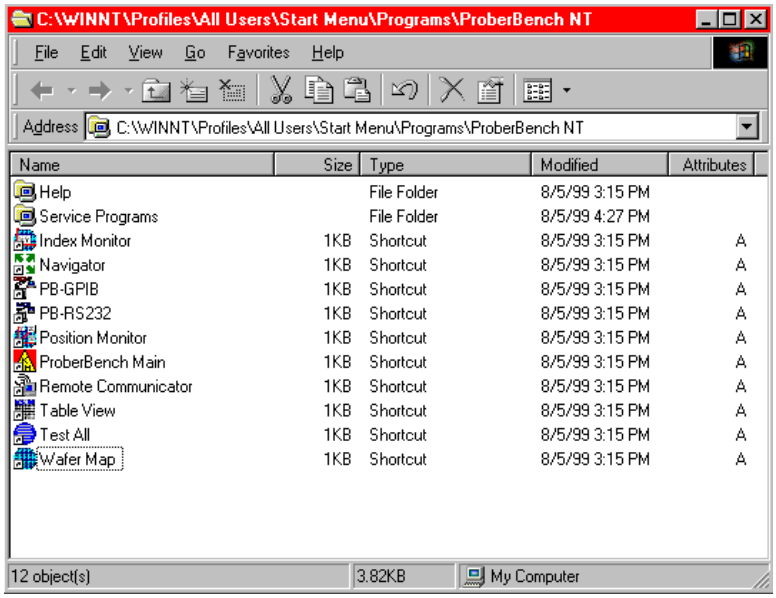
To load a previously defined and saved site definition and a probe list:

1. Select the **ProberBench NT** icon (shortcut) on desktop (Figure H-18).

Figure H-18
ProberBench NT icon

2. From the ProberBench NT window, select **WaferMap** file (Figure H-19).

Figure H-19
ProberBench NT window



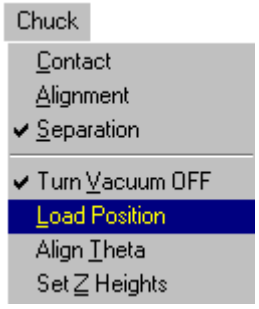
3. From the WaferMap window, select and open the desired file.

Step 4. Load, align, and contact the wafer

NOTE: *The following procedure is accomplished using ProberBench NT computer.*

1. From the WaferMap **Chuck** pull-down, select **Load Position** (Figure H-20). This will bring the chuck to the front of the prober.

Figure H-20
Chuck pull-down

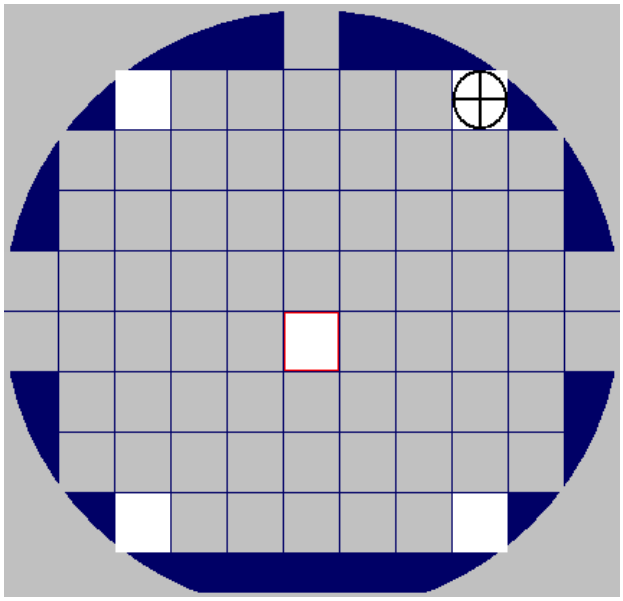


2. Place wafer on chuck.
3. Use chuck pull-down and turn on vacuum.
4. Manually move the wafer to the Home Die.
5. Select **Home Die** from the **Setup** pull-down (Figure H-21).

Figure H-21
Setup pull-down

6. Choose the desired home die on the **WaferMap** (Figure H-22).

Figure H-22
WaferMap home die selection



NOTE: *When choosing the home die:*

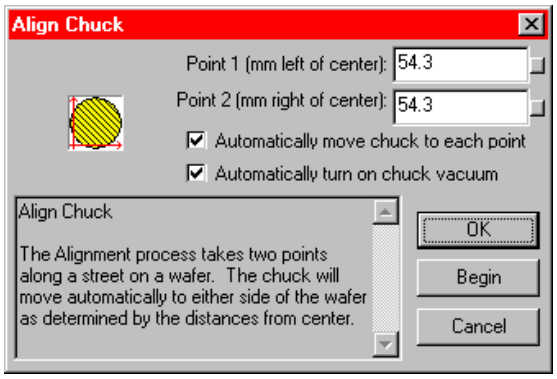
- *The wafer should be on the chuck and physically in the correct HOME position.*
- *Click the die on the wafer map GUI that will be the home die.*
- *A cross-hair appears when a die has been selected as the home die.*

7. Select **Align Theta** from the **Chuck** pull-down.
8. Align wafer using the following four steps:

Step a. Aligning the wafer

- Enter **Point 1** and **Point 2** distances from the center using specific **X** die size multiples (Figure H-23). In other words, if the die size is: X = 13.573mm, and Y = 14.818mm, set up to move four die to the left and also the right at 54.292mm (4 • 13.573mm = 54.292mm).
- Check the following boxes: **Automatically move chuck to each point**; **Automatically turn on chuck vacuum** (Figure H-23).
- Click **Begin**.

Figure H-23
Aligning the wafer: Step a.

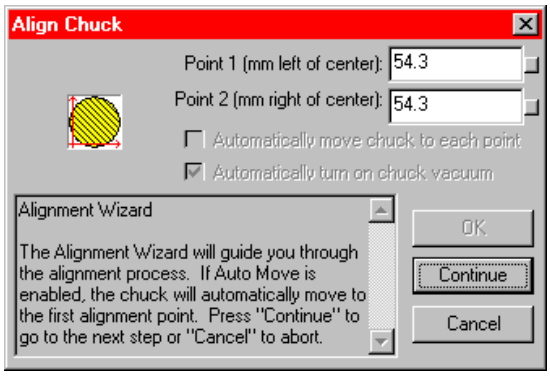


Step b. Aligning the wafer

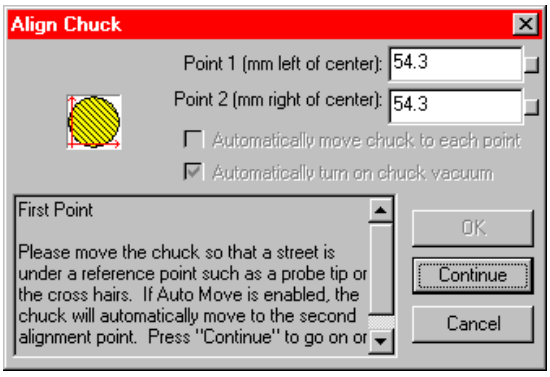
- Move to Point 1 (left of center die align pad and pins) (Figure H-24).
- Select **Continue** (Step b1) to start the Alignment Wizard.
- Manually align pins and pads (POINT 1).
- Select **Continue** (from POINT 1) and move 8 die (for this example) to the right (Step b2).
- Manually align pins and pads (POINT 2) and select finish (Step b3).

Figure H-24
Aligning the wafer: Step b.

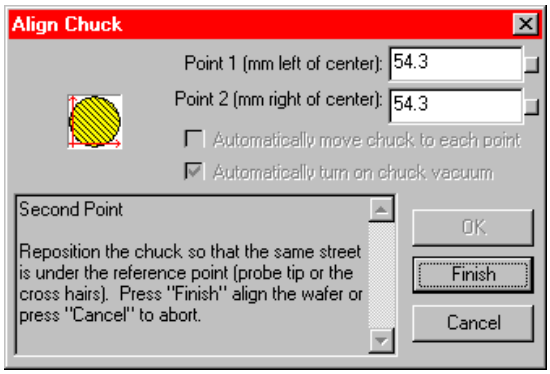
Step b1.



Step b2.



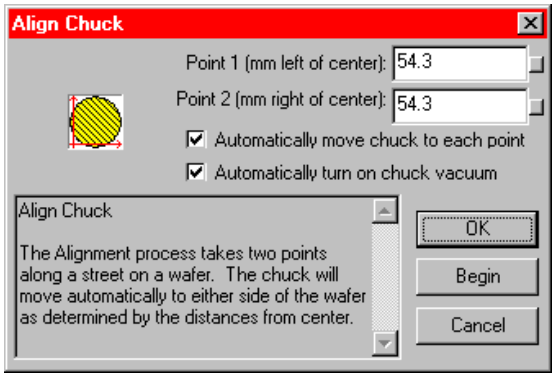
Step b3.



Step c. Aligning the wafer

Confirm that the ta alignment is correct (the alignment procedure is repeated) (Figure H-25). To check, manually use the joystick to move the chuck in index moves and confirm that the pins and pads are aligned.

Figure H-25
Aligning the wafer: Step c.

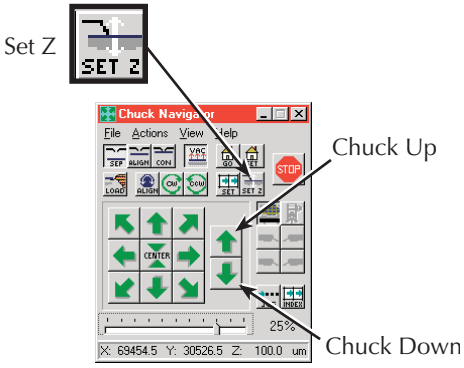


Step d. Aligning the wafer

If the ta alignment is not correct, repeat the four alignment steps in step 1. If the ta alignment is correct, click **Finish** (Figure H-24, Step 2c).

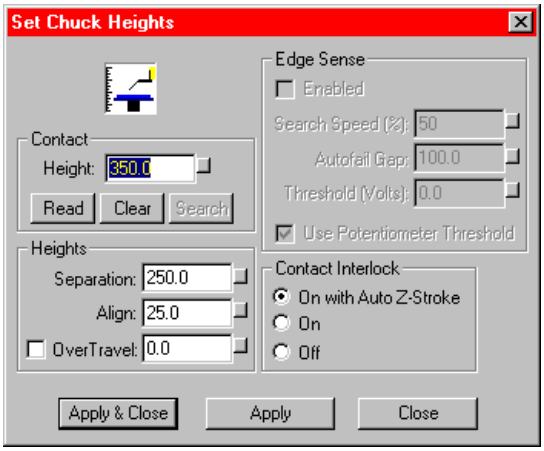
1. Launch the navigator from the ProberBench NT window icon.
2. In the Chuck Navigator window, use the chuck up and down arrows to make contact with the wafer on the desired home die and home subsite (Figure H-26).

Figure H-26
Chuck navigator window: wafer height



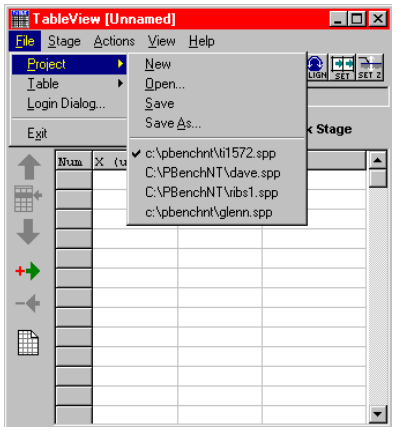
3. Click the **Set-Z** button (Figure H-26). The Set Chuck Height window appears (Figure H-27).

Figure H-27
Set Chuck Heights



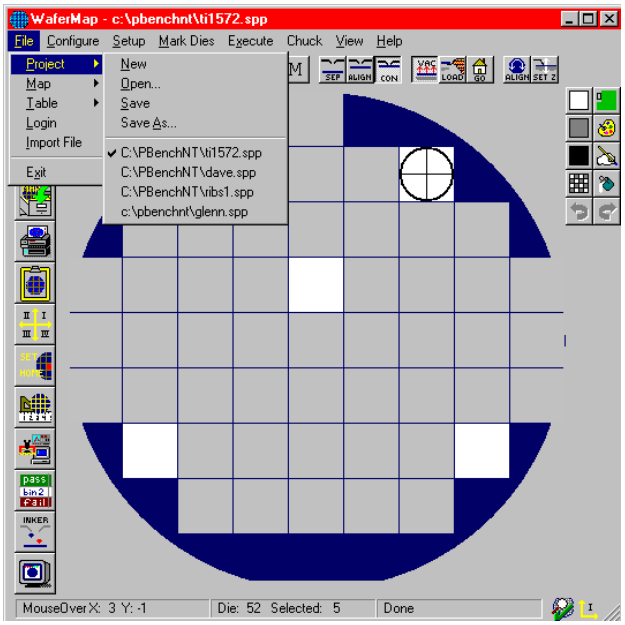
4. Click the **Read** button (contact height value will change to the present height).
5. Click **Apply** and **Close**.
6. **Save** the Chuck Navigator settings (Figure H-28).

Figure H-28
pa200 Chuck Navigator: save



7. **Save** the WaferMap configuration (Figure H-29).

Figure H-29
pa200 WaferMap: save



probesites KITE project example

The following is a step-by-step procedure to properly configure the PA-200 so the **probesites KITE project** executes successfully.

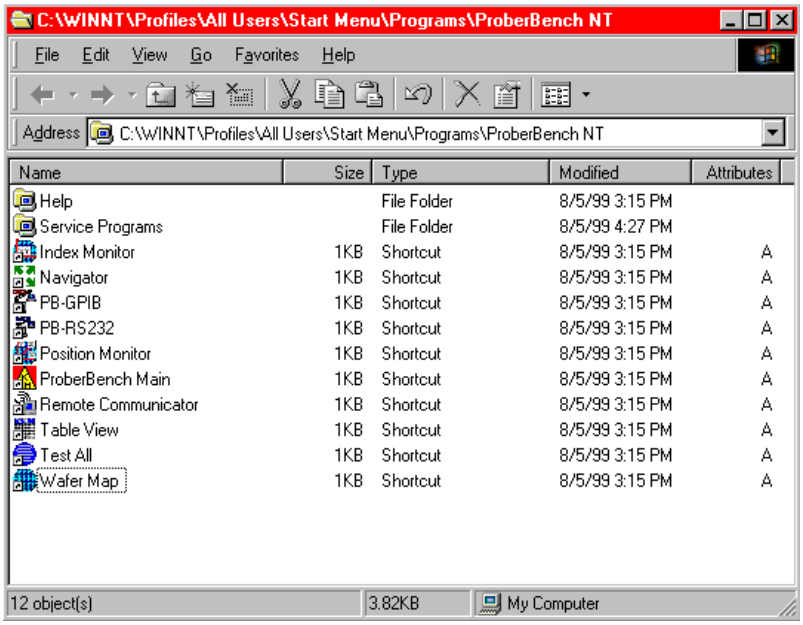
NOTE: *The following configuration is accomplished using the ProberBench NT computer.*

1. Select the **ProberBench NT** icon (shortcut) on desktop (Figure H-30).

Figure H-30
ProberBench NT icon

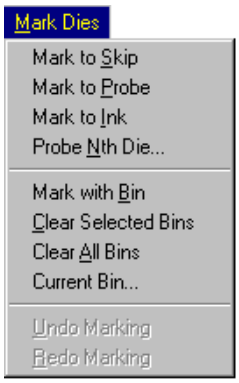
2. From the ProberBench NT window, select **WaferMap** file (Figure H-31).

Figure H-31
ProberBench NT window



3. Use **Mark to Skip** and **Mark to Probe** to set dies as desired (Figure H-32). Clicking on a die in the WaferMap window either sets or clears the die (see note). The die's color indicates status (probes white dies, skips blue dies).

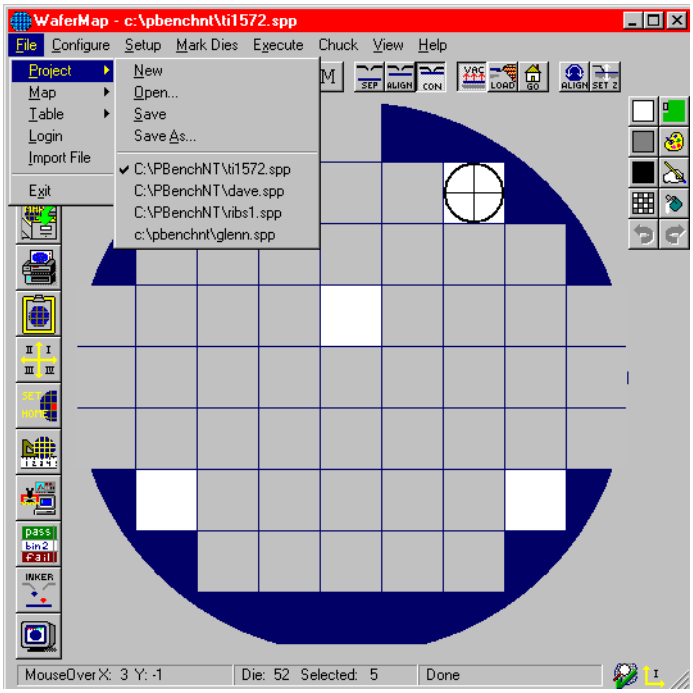
Figure H-32
Mark Dies pull-down



NOTE: *With* **Mark to Probe** *selected, click and drag to select multiple dies. With* **Mark to Skip** *selected, click and drag to clear multiple dies. When done, de-select* **Mark to Skip** *or* **Mark to Probe**. *Otherwise the* **Chuck** *menu will remain grayed.*

4. **Save** the **WaferMap** configuration (Figure H-33).

Figure H-33
pa200 WaferMap: save



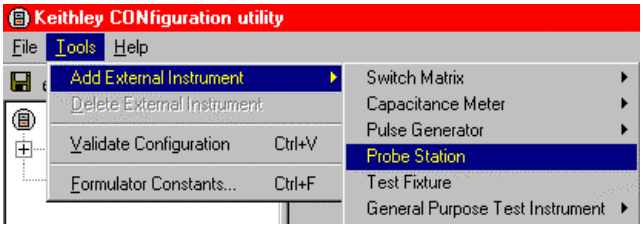
KCON

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KCON to add the prober to the configuration:

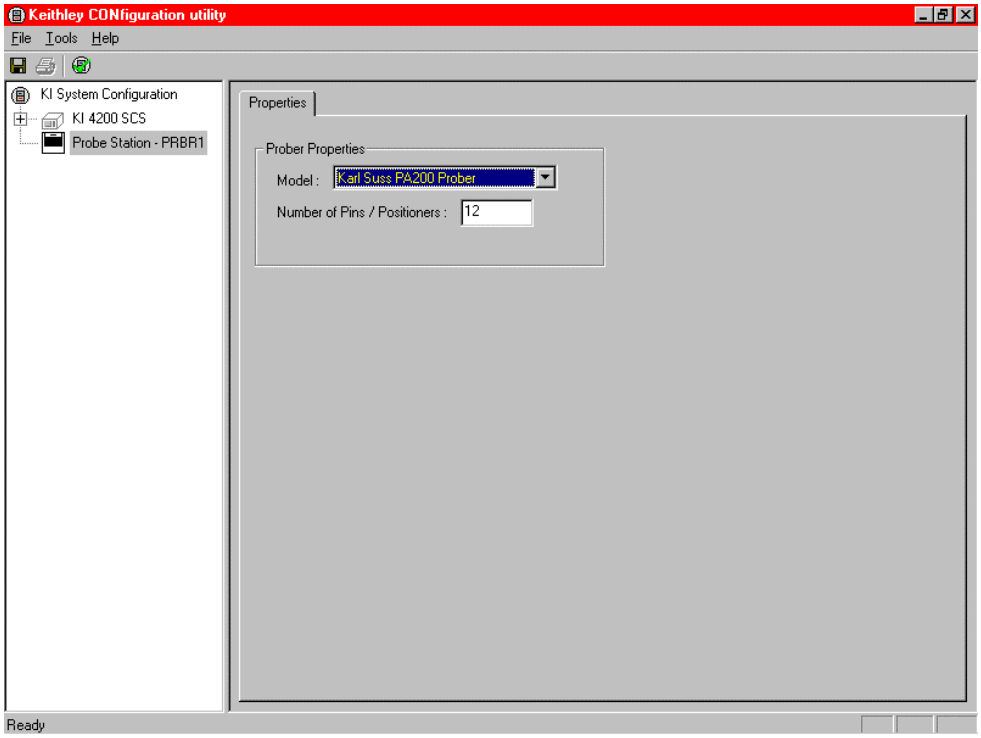
1. From the **Tools** menu (on the Keithley CONfiguration Utility window), click **Add External Instrument > Probe Station** (Figure H-34). The probe station **Properties** tab appears.

Figure H-34
KCON: Adding a prober



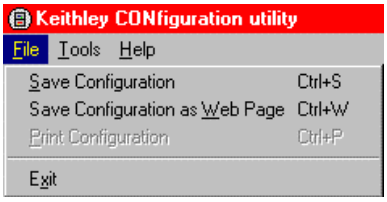
2. Select the Karl Suss prober as the model (Figure H-35). Ensure that the **Number of Pins / Positioners** is correct.

Figure H-35
KCON: Selecting a prober



3. **Save** the configuration from the **File** menu (**Keithley CONfiguration utility** window) (Figure H-36).
4. Exit **KCON**.

Figure H-36
KCON: Saving



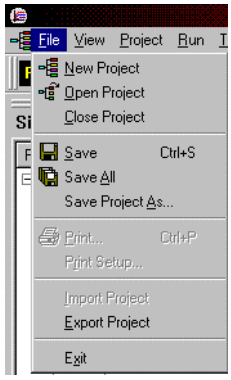
KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KITE to load and run the **probesites** project using the new configuration file, which will allow you to execute the project for this prober.

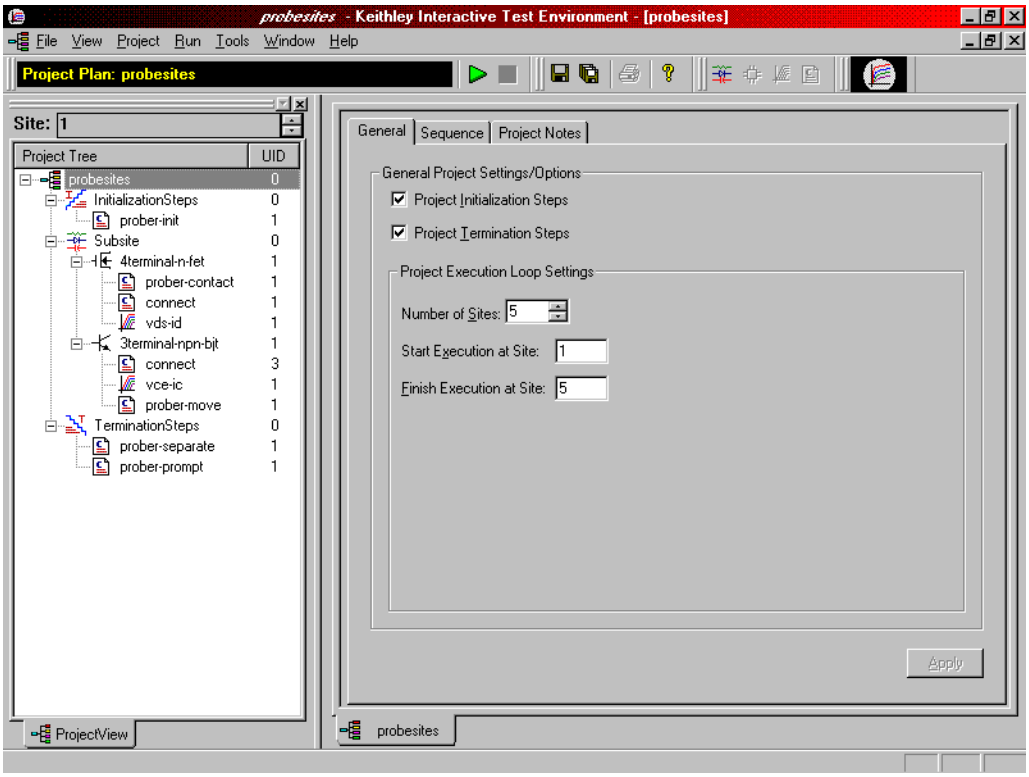
1. Load the **probesites project** example from KITE (Figure H-38):
   - Run **KITE**.
   - Select **Open Project** from the file menu.

Figure H-37
KITE: Open Project



   - Open the folder: `c:\S4200\kiuser\Projects\probesites`
   - Select the project file: **probesites.kpr**

Figure H-38
KITE: probesites project



2. Select the top node in the **ProjectView** tab of the Project Navigator window.
3. Click the green **Run** button.

probesubsites KITE project example

The following is a step-by-step procedure to properly configure the PA-200 so the **probesubsites KITE project** executes successfully.

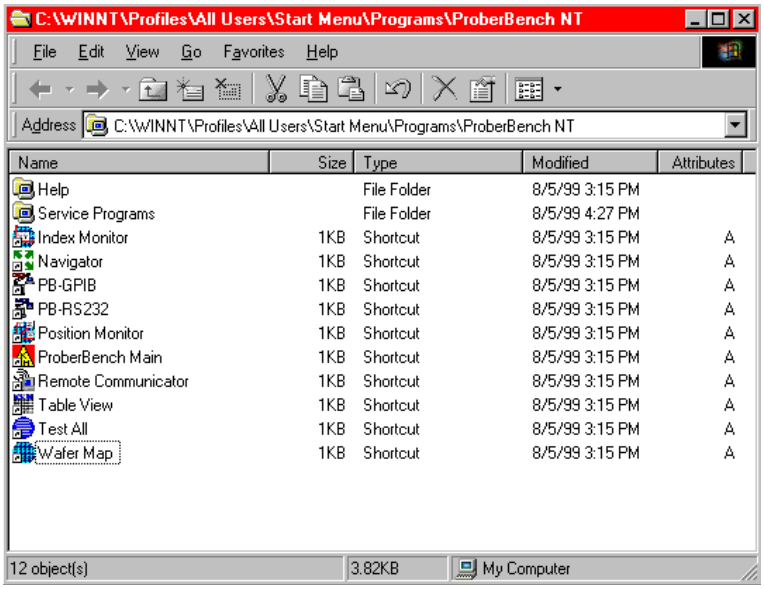
NOTE: *The following configuration is accomplished using the ProberBench NT computer.*

1. Select the ProberBench NT icon (shortcut) on the desktop (Figure H-39).

Figure H-39
ProberBench NT icon

2. From the ProberBench NT window, select **WaferMap** file (Figure H-40).

Figure H-40
ProberBench NT window



3. From the WaferMap window, select **Mark to Skip** from the **Mark Dies** pull-down (Figure H-41).

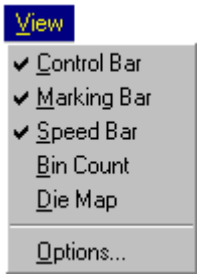
Figure H-41
Mark Dies pull-down

4. Use **Mark to Skip** and **Mark to Probe** to set dies as desired. Clicking on a die in the WaferMap window either sets or clears the die (see **NOTE**). The dies color indicates status (either probe or skip).

NOTE: *With* **Mark to Probe** *selected, click and drag to select multiple dies. With* **Mark to Skip** *selected, click and drag to clear multiple dies. When done, deselect* **Mark to Skip** *or* **Mark to Probe**. *Otherwise the* **Chuck** *menu will remain grayed.*

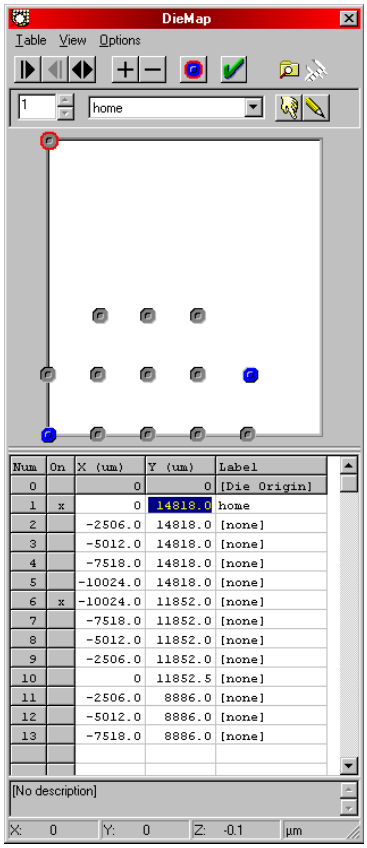
5. Select **Die Map** from the **View** pull-down (Figure H-42). Set up the **die map**:

Figure H-42
View pull-down



- Select **Table editor** from the **View** pull-down in order to display the spreadsheet portion of the **Die Map** (Figure H-43).
- From the **options** pull-down, select units (Microns or Mils).
- Edit the table with the coordinates of the desired subsites.
- Ensure that you save changes using **Save** or **Save As** from the **Table** menu.

Figure H-43
DieMap dialog



NOTE: *An* **x** *in the* **On** *column defines the subsites that will be probed when using subsite probing project (in other words, when using* ***PrSSMovNxt****) even though other subsites may be defined in the list.*

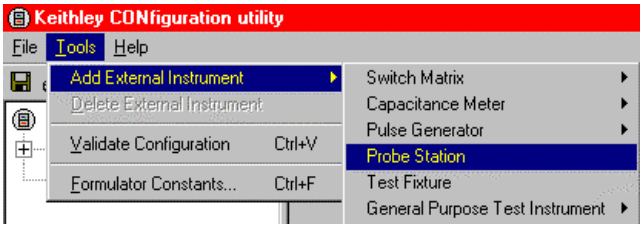
KCON

Use KCON to add the prober to the configuration:

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

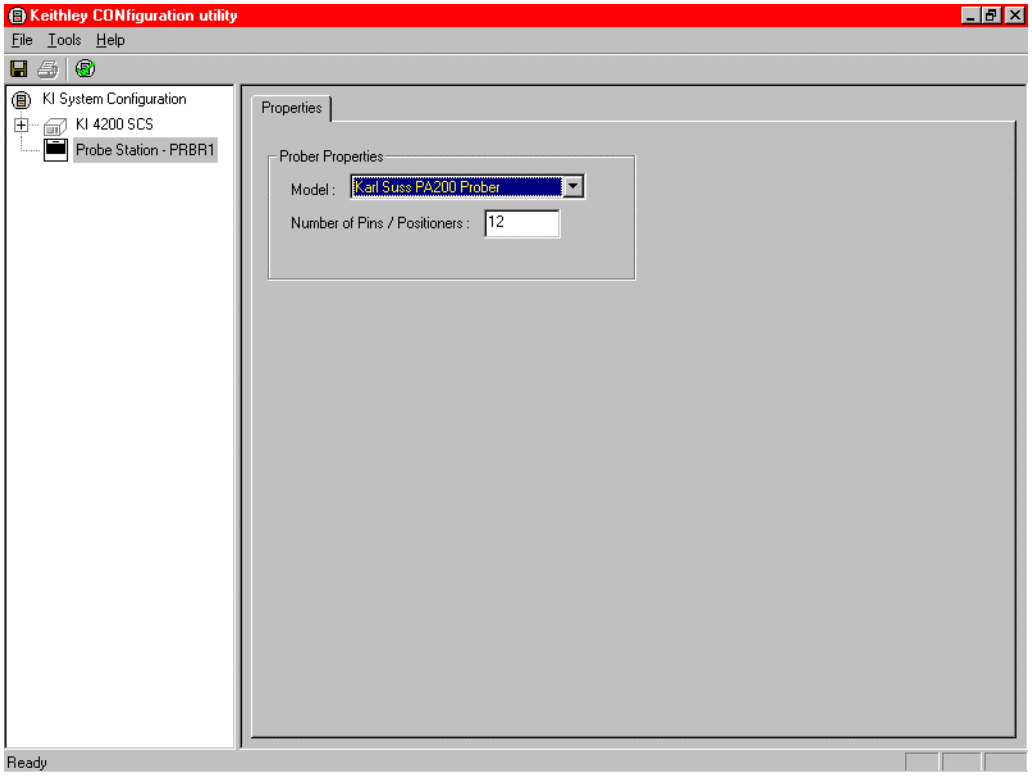
1. From the **Tools** menu (on Keithley CONfiguration Utility window), click **Add External Instrument > Probe Station** (Figure H-44). The probe station **Properties** tab appears.

Figure H-44
KCON: Adding a prober



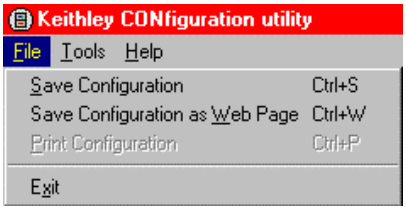
2. Select the Karl Suss prober as the model (Figure H-45). Ensure that the **Number of Pins / Positioners** is correct.

Figure H-45
KCON: Selecting a prober



3. **Save** the configuration from the **File** menu (Keithley CONfiguration utility window) (Figure H-46).
4. Exit **KCON**.

Figure H-46
KCON: Saving



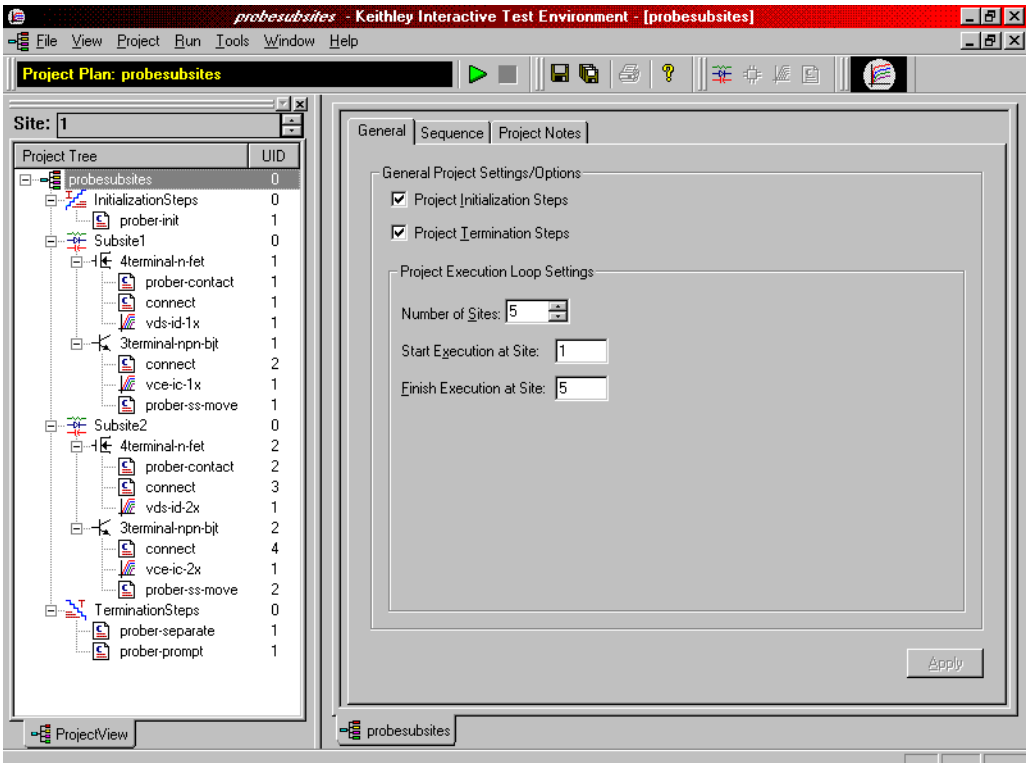
KITE

Use KITE to load and run the **probesubsites project** using the new configuration file which will allow you to execute the project for this prober.

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

1. Load the **probesubsites** project example from KITE (Figure H-47):
   - Run **KITE**.
   - Select **Open Project** from the **File** menu.
   - Open the folder: `c:\S4200\kiuser\Projects\probesites.`
   - Select the project file: **probesites.kpr.**

Figure H-47
KITE: probesubsites project



2. Select the top node in the **ProjectView** tab of the Project Navigator window.
3. Click the green **Run** button.

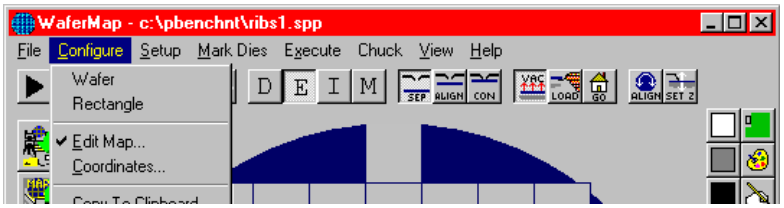
Running projects

After the wafer is set up and alignment is complete, open the **File** menu pull-down in the WaferMap window and:

- Select **Project** and **Save**
- Select **Map** and **Save**
- Select **Table** and **Save**

After saving the project, map, and table, the wafer is ready to probe. Before running the project, place the prober in **Run** mode. Also, ensure that the **"E"** in the **WaferMap** toolbar is selected (Figure H-48).

Figure H-48
WaferMap toolbar



Commands and error symbols

The following list (Table H-2) contains error and status symbols listed by command.

Table H-2
Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt
PR_OK	X	X	X	X
BAD_CHUCK	X			
INVAL_MODE	X			
UNINTEL_RESP	X	X	X	X
INVAL_PARAM		X		
BAD_MODE		X	X	X
UNEXPE_ERROR		X	X	X
PR_WAFERCOMPLETE			X	X

Appendix I
Micromanipulator 8860 Prober

In this section:

Required probe station software

The following six programs are used to configure and operate the 8860 prober with the Keithley Instruments Model 4200-SCS (all are required):

NOTE: *To properly use the 8860 prober with the Model 4200-SCS, software programs pcIndie and pcWafer (not included with standard prober software) are required. Refer to the prober manufacturer, Micromanipulator, for availability.*

- **pcBridge**: Used to configure the communications setup (icon located on the desktop)
- **pcLaunch**: Used to launch various wafer controls and utilities (icon located on the desktop)
- **pcIndie**: Used to probe multi-subsites per die (button located in pcLaunch window)
- **pcWafer**: Used to probe single subsites per die (button located in pcLaunch window)

Software versions

The following list contains the software versions used to verify the configuration of the 8860 prober with the Model 4200-SCS:

Product Name: pcbridge
Product Version: 2.0.2

Product Name: pcIndie
Product Version: 2.0.7

Product Name: pcLaunch
Product Version: 2.0.9

Product Name: pcNav
Product Version: 2.0.8

Product Name: pcWfr
Product Version: 2.0.8sProduct Name: pcRouter
Product Version: 2.0.9

Probe station configuration

CAUTION **Although this appendix provides instructions on prober setup and configuration, ensure that you are familiar with the Micromanipulator 8860 prober and its supporting documentation before attempting setup, configuration, or operation.**

There are four general steps required to set up and configure the 8860 prober for use with the Model 4200-SCS:

Step 1. Setting up communication

Step 2. Setting up wafer geometry

Step 3. Creating a site definition and defining a probe list

Step 4. Loading, aligning, and contacting the wafer

Each step is detailed later in this section.

Modifying the prober configuration file

NOTE: *This file is modified using the Model 4200-SCS computer.*

The default prober configuration file is contained in Figure I-1. As shown, the file is configured for use with a serial prober setup.

Configuration file location: `C:\S4200\sys\dat\prbcnfg_MM40.dat`

Figure I-1
Sample 8860 prober configuration file

```
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS ""== NULL, max 32 chars in string
#
# Example
#        01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#   OcrPresent
#   AutoAlnPresent
#   ProfilerPresent
#   HotchuckPresent
#   HandlerPresent
#   Probe2PadPresent
#
#
# The PROBER_x_PROBTYPE fields needs to be set to one of the following names.
# Configuration for serial probers:
#
#  Example configuration for MM40 prober
#
#
PROBER_1_PROBTYPE=MM40
PROBER_1_OPTIONS=0,0,0,0,0,0
PROBER_1_IO_MODE=SERIAL
PROBER_1_DEVICE_NAME=COM1
PROBER_1_BAUDRATE=9600
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
#
```

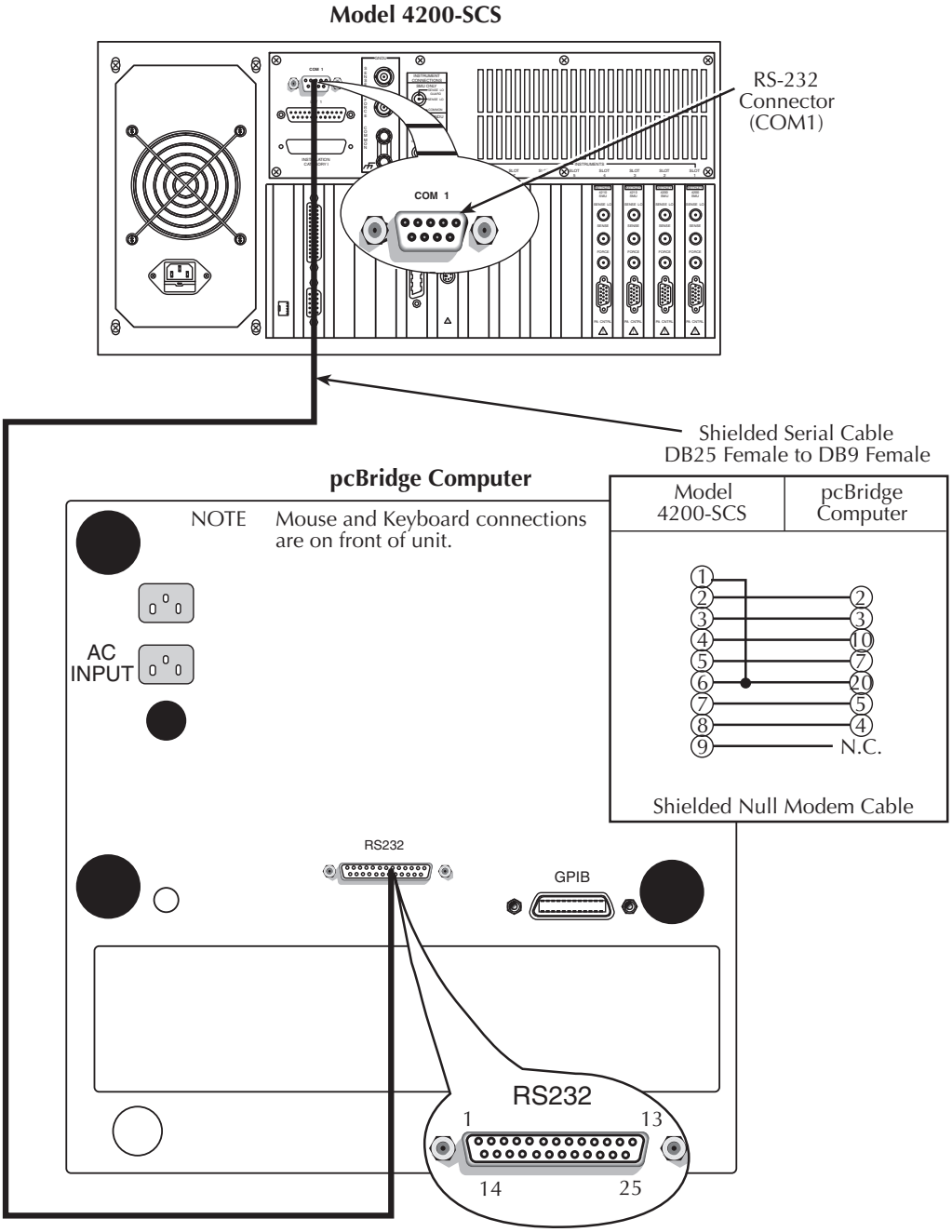
Step 1. Setting up communication

1. Turn on Model 4200-SCS power.
2. Turn on power to the prober.
3. Ensure that the vacuum has been properly connected.

NOTE: *The following configuration is accomplished using the pcBridge computer.*

4. Connect the pcBridge computer's RS232 port (located on the rear panel of the pcBridge computer) to the Model 4200-SCS COM1 port using a DB25 female to DB9 female cable (shielded null modem cable). Refer to Figure I-2.

Figure I-2
Prober setup: Serial connections



5. Double-click the **pcBridge** icon (shortcut) on desktop (Figure I-3).
6. From the pcBridge window (Figure I-4), open the **Setup** pull-down menu. The pcBridge Communications Setup window will appear (Figure I-5).
7. Use the pcBridge Communications Setup to configure the communication settings:
   - Interface Type: RS232
   - Baud: 9600
   - Port: COM2
   - Term: cr and lf (termination character of carriage-return and line-feed)

   Settings should be 8 data, 1 stop, no parity, xon / xoff.
8. Click **OK**.

Figure I-3
Prober setup: pcBridge icon

Figure I-4
Prober setup: pcBridge window (main)

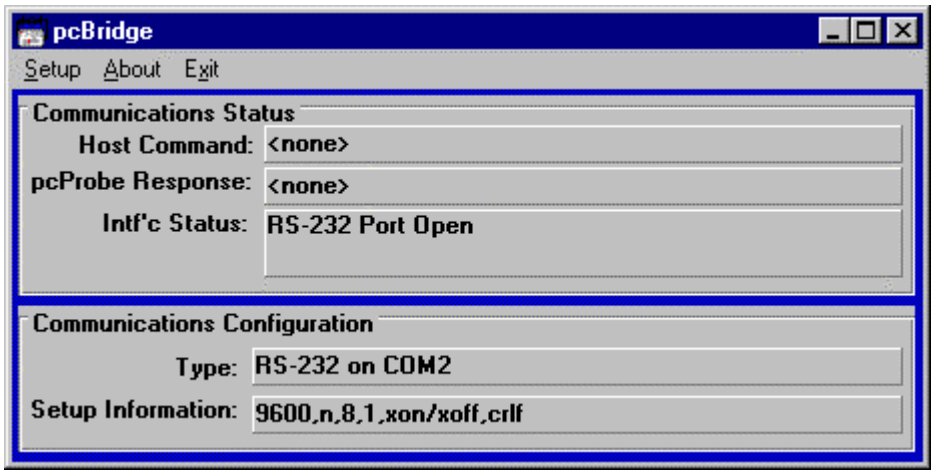
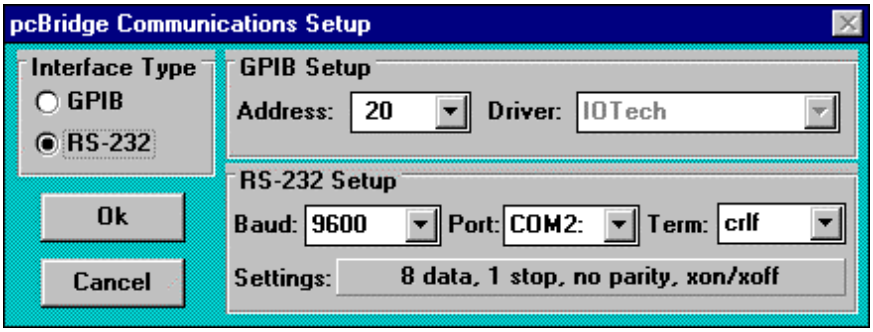


Figure I-5
Prober setup: pcBridge Communications Setup window



9. Start pcLaunch by clicking the **pcLaunch** icon (Figure I-6). The pcLaunch window will appear (Figure I-7).
10. From the pcLaunch window, set the **Joystick Mode** for **Linear** (Figure I-8).

Figure I-6
Pclaunch icon

Figure I-7
pcLaunch window

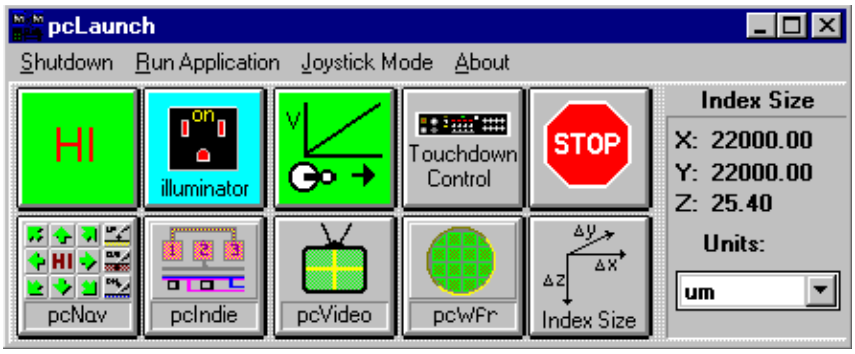
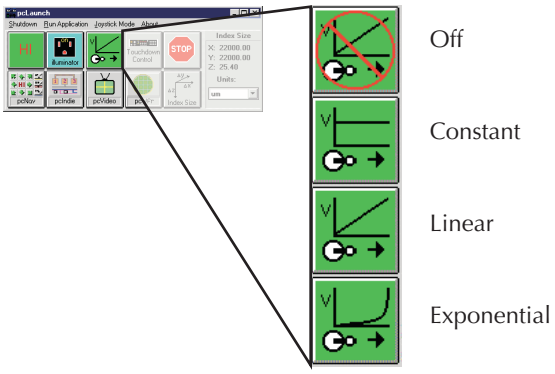


Figure I-8
Joystick modes



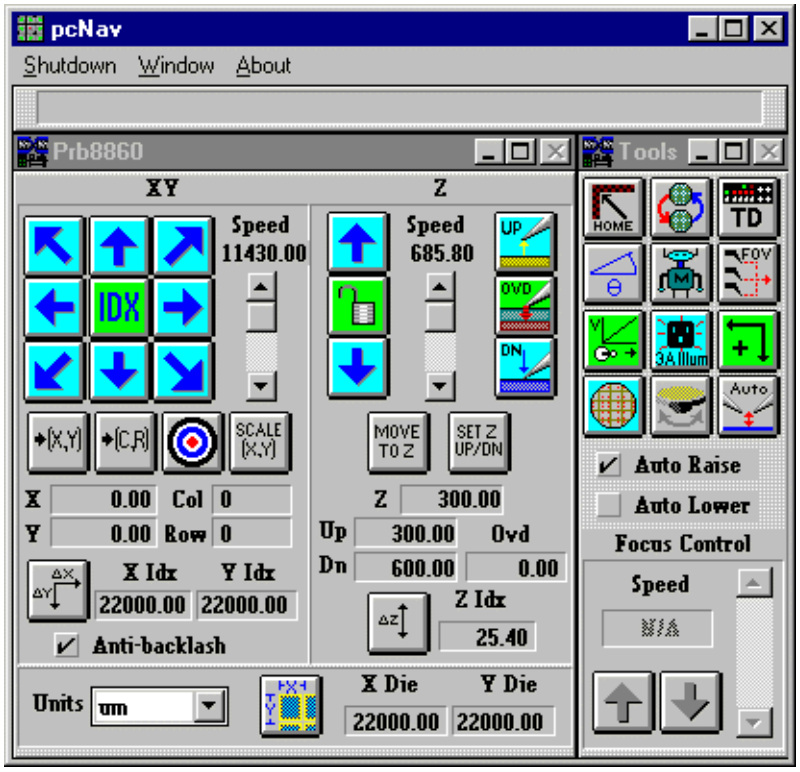
Step 2. Setting up wafer geometry

NOTE: *The following configuration is accomplished using the pcBridge computer.*

1. From the pcLaunch window, click the **pcNav** button (Figure I-9). The pcNav window opens (Figure I-10).

Figure I-9
pcNav button

Figure I-10
pcNav window



NOTE: *When starting pcNav for the first time, the warning contained in Figure I-11 will appear. Click* **OK** *and continue the configuration (the device will be initialized when the chuck is homed).*

Figure I-11
pcNav Boot warning

NOTE: *Since the platen moves to make or break contact between the pins and pad, clicking **Auto Raise** will automatically separate the pins from the pads while **Auto Lower** will allow automatic contact.*

2. Check **Auto Raise** on the pcNav Tools window.
3. Check **Anti-backlash** on the pcNav Prb8860 window.

Step 3. Creating a site definition and defining a probe list

NOTE: *The following setup procedure is accomplished using the pcBridge computer.*

Creating a site definition for a single subsite per die involves using the software to create a selection of dies to probe. If a single subsite per die is to be probed, refer to probesites KITE project example later in this appendix. Creating a site definition for multiple subsites per die also involves using the software to create a selection of dies to probe, but also includes creating a selection of the subsites on each die that will be probed. If multiple subsites per die will be probed, refer to probesubsites KITE project example later in this appendix.

Use the following information to load a previously-defined and saved site definition.

Single subsite per die

1. Start pcWafer by clicking the **pcWfr** button (Figure I-12) located in the pcLaunch window. The pcWfr window will appear.
2. Click the **Open** button on the Die Program Tools window (Figure I-13) to open an existing file or **New** to create a new file.
3. Select the desired file, and click **OK**.

Figure I-12
pcWfr button

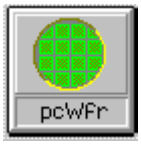
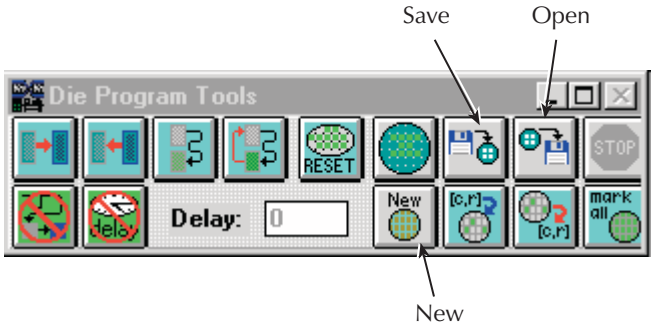


Figure I-13
Die Program Tools window



Multiple subsites per die

4. Click the **pcIndie** button (Figure I-14) located in the pcLaunch window. The pcIndie window will appear.
5. Click the **Open** button on the pcIndie Edit window (Figure I-15) to open an existing file or **New** to create a new file.
6. Select the desired file, and click **OK**.

Figure I-14
pcIndie button

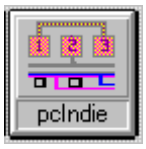
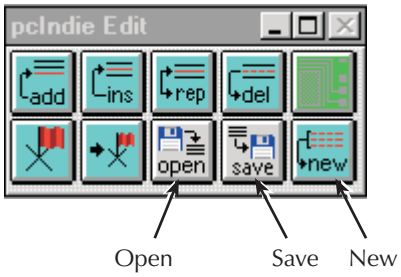


Figure I-15
pcIndie Edit window



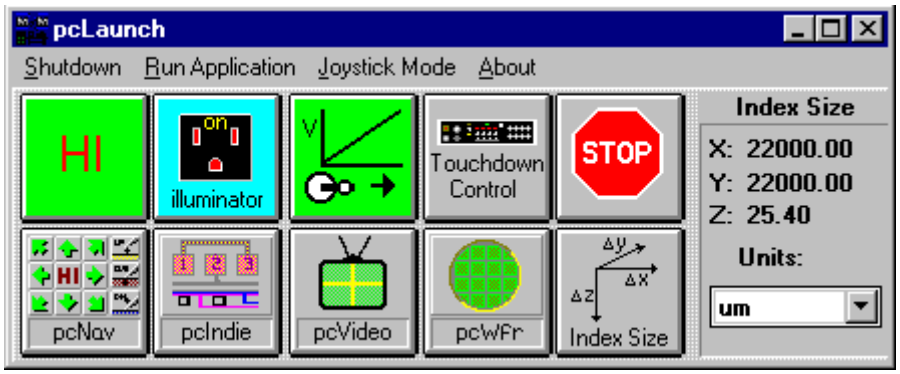
Step 4. Loading, aligning, and contacting the wafer

NOTE: *The following procedure is accomplished using the pcBridge computer.*

1. Start pcLaunch by clicking the **pcLaunch** icon (Figure I-16). The pcLaunch window (Figure I-17) will appear.

Figure I-16
pclaunch icon

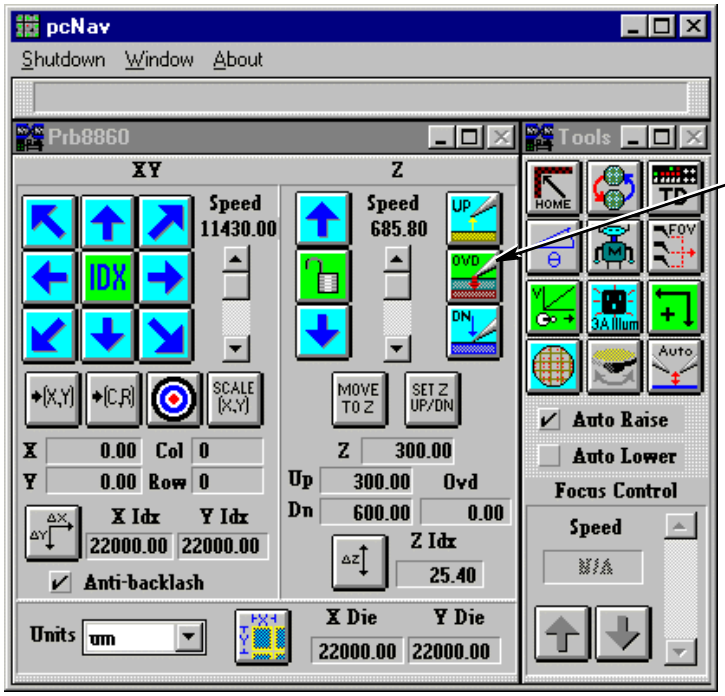
Figure I-17
pcLaunch window



2. Home chuck:
   - From the pcLaunch window, click the **pcNav** button (Figure I-18). The pcNav window opens (Figure I-19).
   - Click the **Home** button (Figure I-20) on the **Tools** panel of the pcNav window. The Initialize positioners to Home window opens (Figure I-21).
   - From the Initialize positioners to Home window, click the **Home chuck** button. The chuck moves to the back left corner and then to the middle.
   - Click the **Done** button when chuck is home (the **Done** button will turn from grayed to active when the chuck is home).

Figure I-18
pcNav button

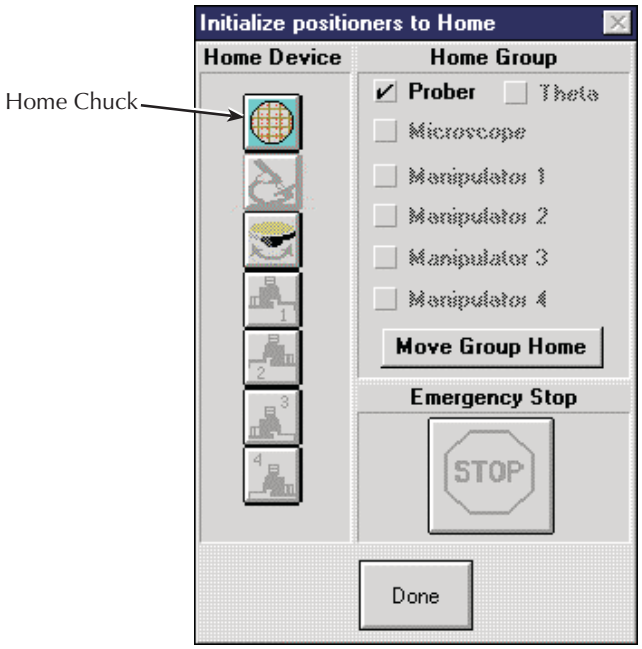
Figure I-19
pcNav window



NOTE: *OVD button toggles the state of the overdrive (on or off).*

Figure I-20
Home button

Figure I-21
Initialize positioners to Home window



3. Ensure the vacuum is off.
4. Click the **Load wafer** button on the Tools panel of the pcNav window (Figure I-22). A **Load Wafer** dialog box appears (Figure I-23).
5. In the **Load Wafer** dialog box, click **Load**.
6. After the chuck moves to the front, place wafer on the chuck aligning the flat or notch in the proper orientation.
7. Apply vacuum.
8. Click **Center** (Figure I-23).
9. Click **Done**.

NOTE: *This part of the procedure sets Z-height (contact height).*

The platen moves up and down (Z) while the chuck moves X and Y but not Z. When changing Z-height (moving the platen up or down), a higher number moves closer to contact while a lower number moves away from contact (for example, if 300 is contact, 200 would be non-contact).

10. Use the joystick to manually move the wafer (pads) underneath the pins.

Figure I-22
Load Wafer button

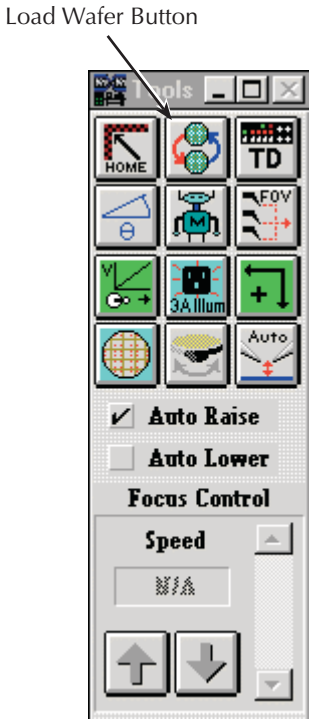
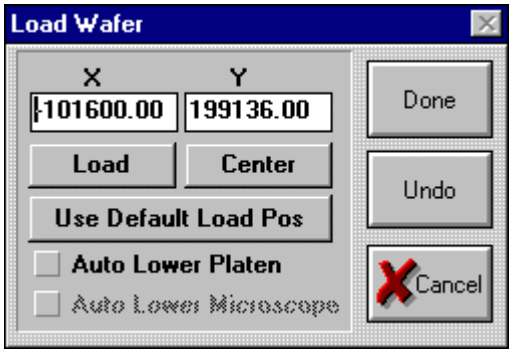


Figure I-23
Load Wafer window



11. Click the **SET Z UP/DN** button on pcNav window (Figure I-24). The SET Prb8860 Up/Down/Ovd window will open (Figure I-25).
12. Using the Dial, bring the platen to a positive Z-height (this height in the example is 600). This will be a non-contact position with the pads in focus but without the pins touching the pads (Figure I-27).
13. Use the manual Z-dial to lower the platen to make initial contact with pads (this assumes that the pins are planar).
14. Click the **Set Down** button when all pins are in contact with their respective pad.
15. Use the Dial to move the pins to a non-contact position (this height in the example is 300).
16. Click the **Set UP** button.

NOTE: *If the pins are not aligned to the same plane, excessive overdrive / scrub may result (overdrive is the Z-height change necessary to exert adequate contact pressure on*

the pad). Keep this as equal as possible when manually setting the pins on the pads. Using uneven contact pressure to overcome planarization problems can cause faulty test results or damage to the pad.

17. Set overdrive (user preference).
    a. Click the **DN** button (pcNav window) then press the **Set Z UP/DN** button (Figure I-24). Once the SET Prb8860 Up/Down/Ovd window opens (Figure I-25), press the **Set Base Pt** button.
    b. Lower the platen to the point where you see good clean probe marks. At this point, click the **Set 2nd Pt** button (Figure I-25).
    c. Click the **OVD** button in pcNav to Ensure that the overdrive will be used (Figure I-19). Test the settings by pressing the UP and DN buttons (Figure I-26) in pcNav.
18. Click **Done**.

Figure I-24
Set Z UP/DN button

Figure I-25
SET Prb8860 Up/Down/Ovd window

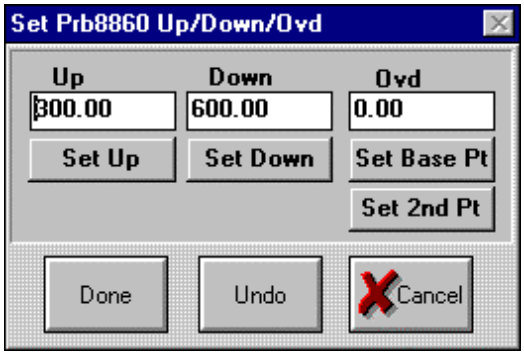
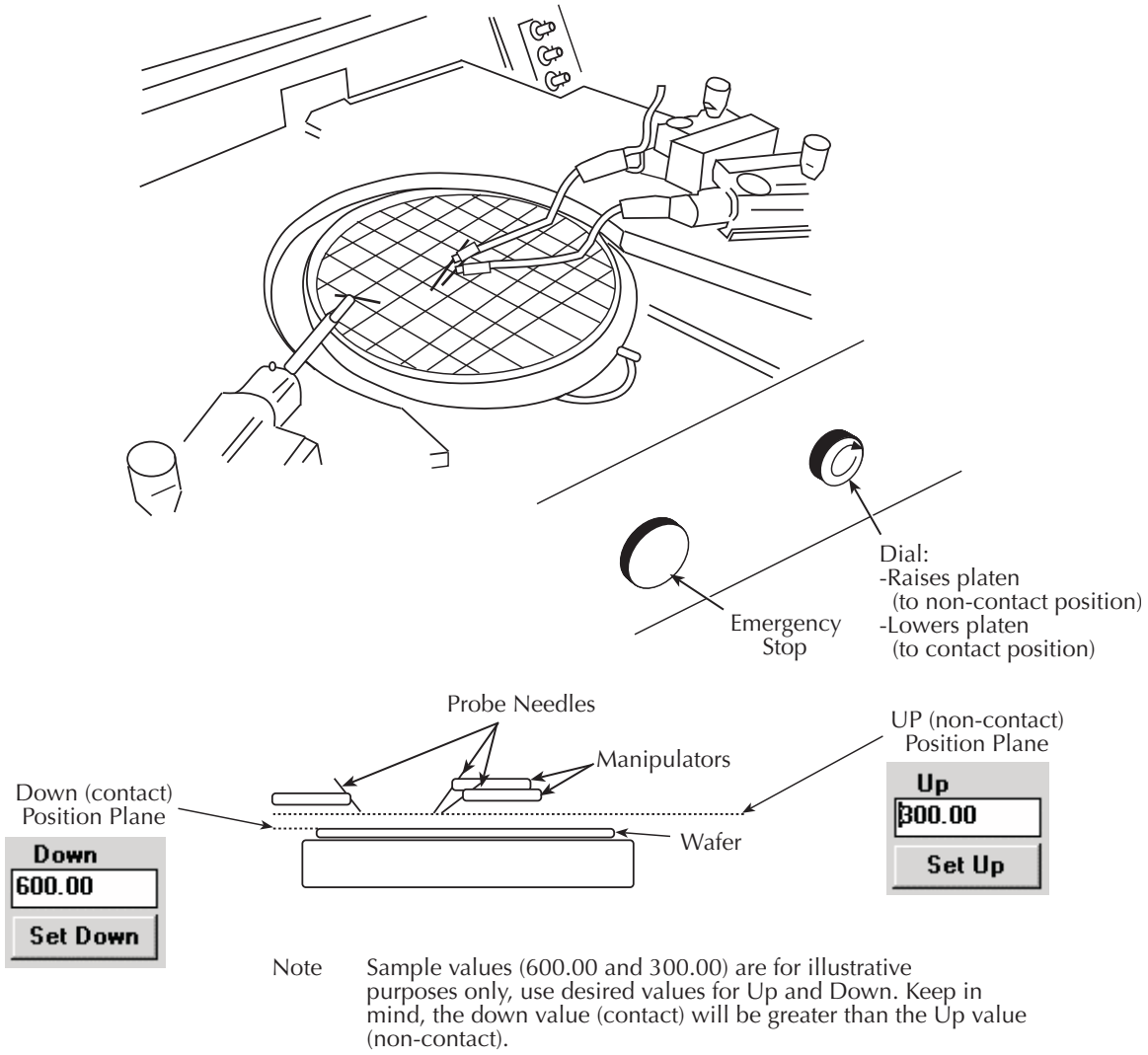


Figure I-26
Down pushbutton

Figure I-27
Set Z-height



Note Sample values (600.00 and 300.00) are for illustrative purposes only, use desired values for Up and Down. Keep in mind, the down value (contact) will be greater than the Up value (non-contact).

NOTE: *This part of the procedure aligns the wafer.*

19. Click the **align wafer** button (Figure I-28) on the **Tools** panel of the pcNav window. The Prb8860 Alignment window will open (Figure I-29).

Figure I-28
Align wafer button

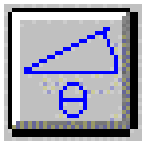
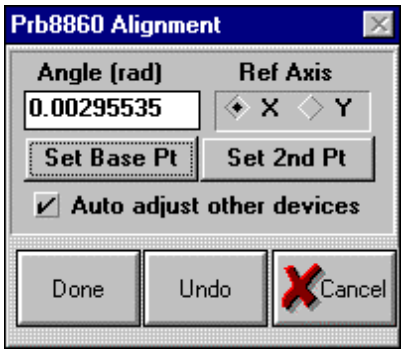


Figure I-29
Prb8860 Alignment window



20. Select **Ref Axis X** in the Prb8860 Alignment window.
21. Check **Auto adjust other devices**.
22. Move prober chuck to extreme left of the wafer. Look through the microscope and Ensure the pins are over the pads.
23. Click **Set Base Pt**.
24. Move to a die on the extreme right of the wafer.
25. Use the joystick (low mode) and theta adjustment to align the pins to the same pads as the first die (both along the same row of die).
26. Click **Set 2nd Pt**.
27. Repeat this process until the Angle (rad) is as close to zero as possible.
28. When the alignment is complete, click **Done**.

NOTE: *This part of the procedure sets units and die size.*

29. Set units to either microns or mils from pcNav's Prb8860 window (lower left corner).

Figure I-30
Unit of measure drop-down list box

30. Click **Set X, Y die size** button located in the lower middle of the Prb8860 window. The **Set X, Y Die Size** dialog box opens. If die size is known, enter it. If not known, calculate (see Calculating die sizes in the following text).

Figure I-31
Set X, Y die size button

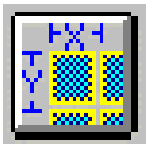
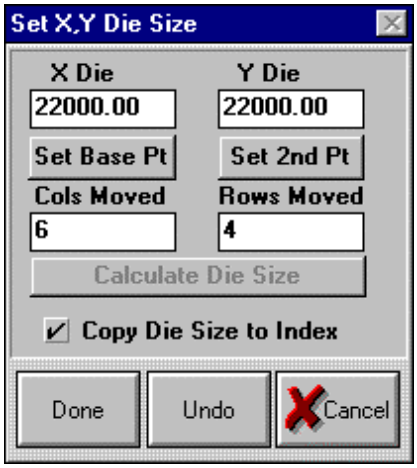


Figure I-32
Set X, Y Die Size dialog box



Calculating die sizes

a. Place pins over pads in upper left corner of wafer (although the upper left corner die is used in this example, any die may be selected as a base point).
b. Click **Set Base Pt**.
c. Move over and down a known number of die; enter these values into Column moved (columns) and Row moved (rows).
d. Click the **Set 2nd Pt** button.
e. Check the **Copy Die Size to Index** button.
f. Click the **Calculate Die Size** button.

This determines the accurate die size. To complete, click **Done** (the grayed out button will turn active after the calculation is completed).

31. Click the **Set Reference Die** button from the pcNav window. The **Set Reference** dialog box opens (Figure I-34).

Figure I-33
Set Reference Die button

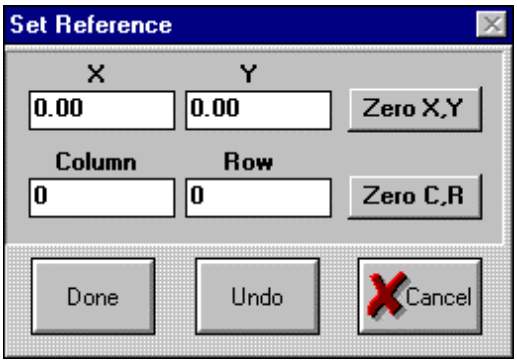
32. Move over the pads of the reference die (reference die is user-selected).
33. Zero out the X/Y and Column and rows (click **Zero X,Y** button and **Zero C,R**).

NOTE: *If you want the columns and rows to be something other than 0,0 (1,1 for instance), edit values in* **Set Reference** *dialog box as desired before clicking the* ***Done*** *button.*

---

34. Click **Done**.

Figure I-34
Set Reference dialog box



probesites KITE project example

The following is a step by step procedure to properly configure the 8860 so the **probesites KITE project** executes successfully.

NOTE: *The following configuration is accomplished using the pcBridge computer.*

Use the pcWafer program to probe a single subsite on multiple dies.

1. Start pcWafer by clicking the **pcWfr** button located in the pcLaunch window. The pcWfr window will appear.

Figure I-35
pcWfr button

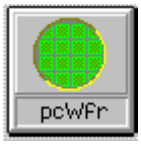
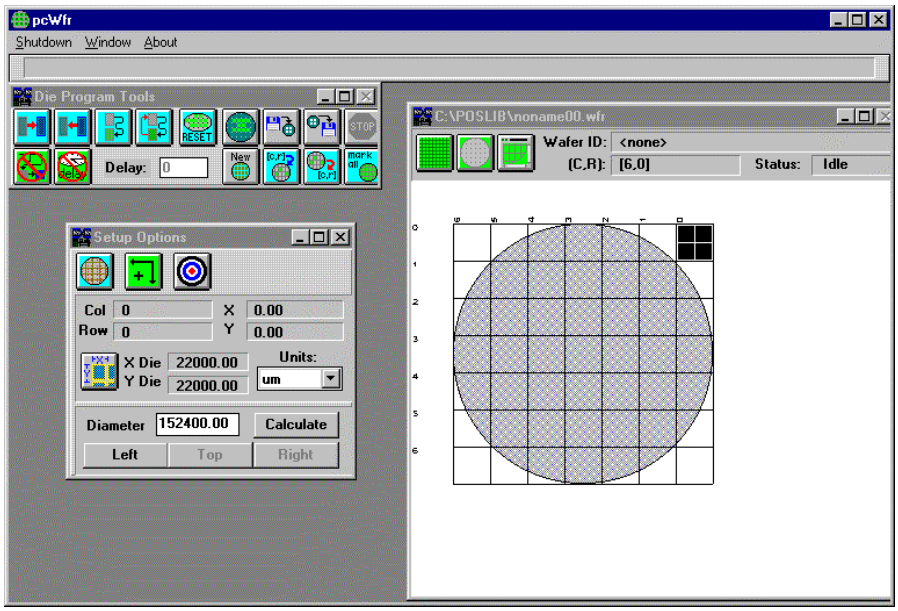


Figure I-36
PcWfr window



2. Set units of measure (microns or mils).
3. Calculate the wafer diameter:
   a. Move the pins to the left edge of the wafer.
   b. Click **Left** button on the Setup Options window.
   c. Repeat for the Top and Right edges of the wafer, clicking the respective buttons after each movement.
   d. Click the **Calculate** button.
4. Set units to either microns or mils from the unit of measure **Units** drop-down list box (Figure I-37) located in Setup Options window (Figure I-38).

Figure I-37
Unit of measure drop-down list box

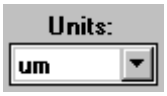
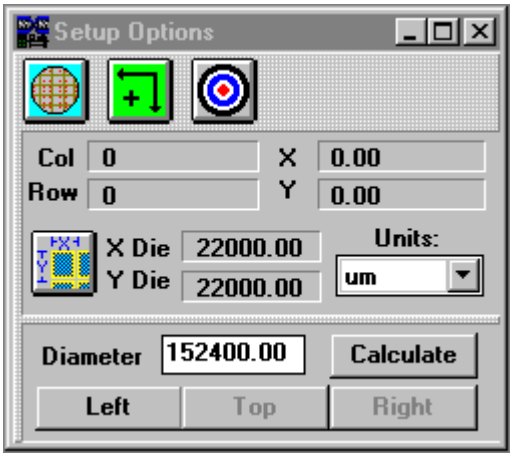


Figure I-38
Setup Options window



5. Click **Set X, Y** die size button located in the Setup Options window. The Set **X, Y Die** size dialog box opens.
6. Set **X, Y Die** size button.
7. If die size is known, enter it. If not known, calculate (see Calculating die sizes in the following text).

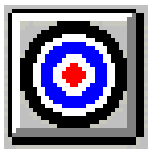
Calculating die sizes

a. Place pins over pads in upper left corner of wafer (although the upper left corner die is used in this example, any die may be selected as a base point).
b. Click **Set Base Pt**.
c. Move over and down a known number of die, enter these values into Column moved (columns) and Row moved (rows).
d. Click the **Set 2nd Pt** button.
e. Check the **Copy Die Size to Index** button.
f. Click the **Calculate Die Size** button.

This determines the accurate die size. To complete, click **Done** (the grayed out button will turn active after the calculation is completed).

8. Click the **Set Reference Die** button located in the **Setup Options** window (Figure I-39). The **Set Reference** dialog box opens.

Figure I-39
Set Reference Die button

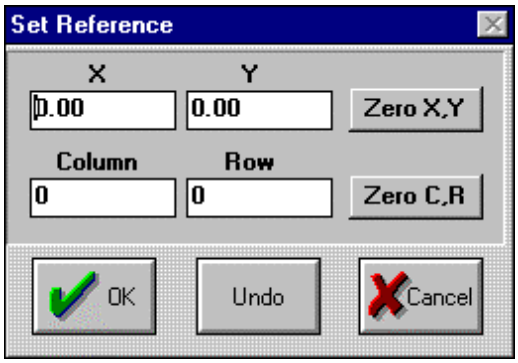


9. Move the pins over the pads of the die desired to be set as reference die.
10. Zero out the **X/Y**, **Column**, and **Row** (click **Zero X,Y** button and **Zero C,R**).

NOTE: *If you want the columns and rows to be something other than 0,0 (1,1 for instance), edit values in* **Set Reference** *dialog box as desired before clicking the* **Done** *button.*

11. Click **Done**.

Figure I-40
Set Reference dialog box



12. Select the dies to test using the mouse (site turns green to test).

NOTE: *The order of selection of the die, the spline pattern (change using edit die program), and the reference die location determine test order sequence.*

---

13. Set spline pattern (optional).
    - Click the **Edit Die Program Parameters** button located on the Die Program Tools window of pcWfr. The **Edit Die Program Parameters** dialog box will open (Figure I-41).
    - Click the **Spline Pattern** button on the **Edit Die Program Parameters** dialog box. The Spline Pattern window will open (Figure I-43).
    - Select desired spline pattern (the icon of the active spline pattern is transferred to the **Edit Die Program Parameters** dialog box).

Figure I-41
Die Program Tools window

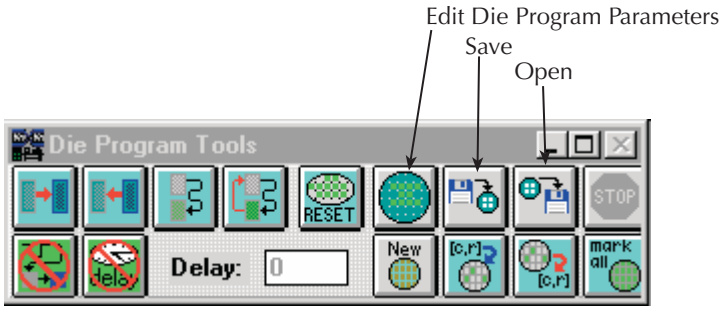


Figure I-42
Edit Die Program Parameters window

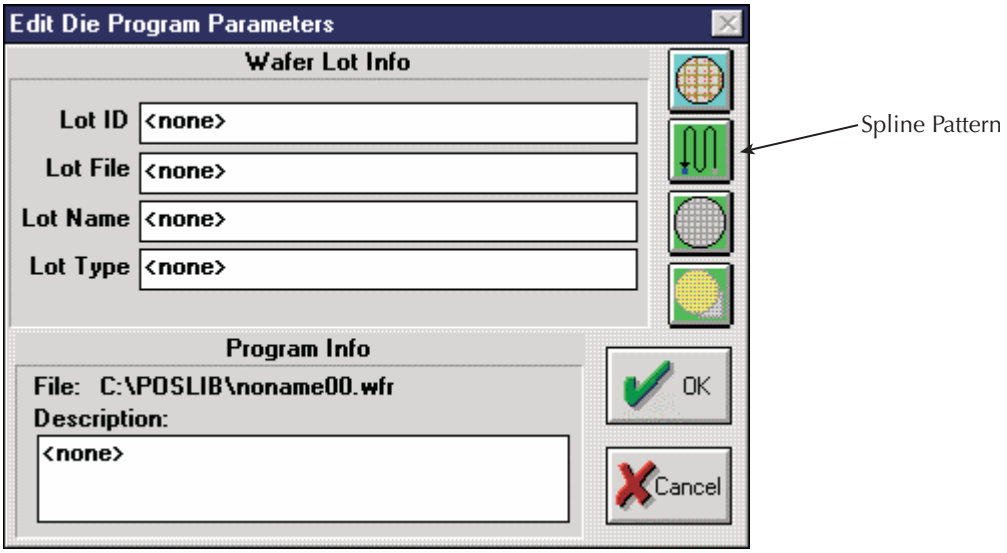
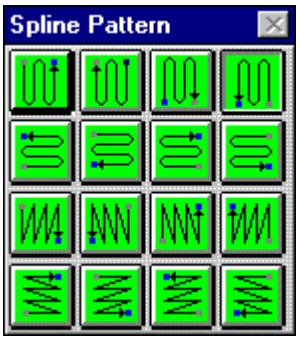


Figure I-43
Spline Pattern window



14. Click the **Save** button on the Die Program Tools window (Figure I-41).
15. To open an existing program listing file, click the **pcWfr Open** button on the Die Program Tools window. Select the desired file, and click **OK**.

NOTE: *Before starting testing, physically align the pins over the reference die.*

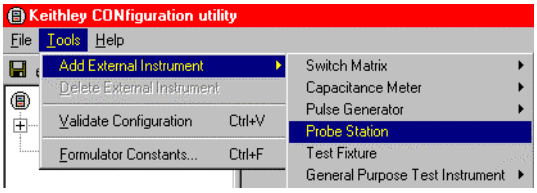
KCON

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KCON to add the prober to the configuration:

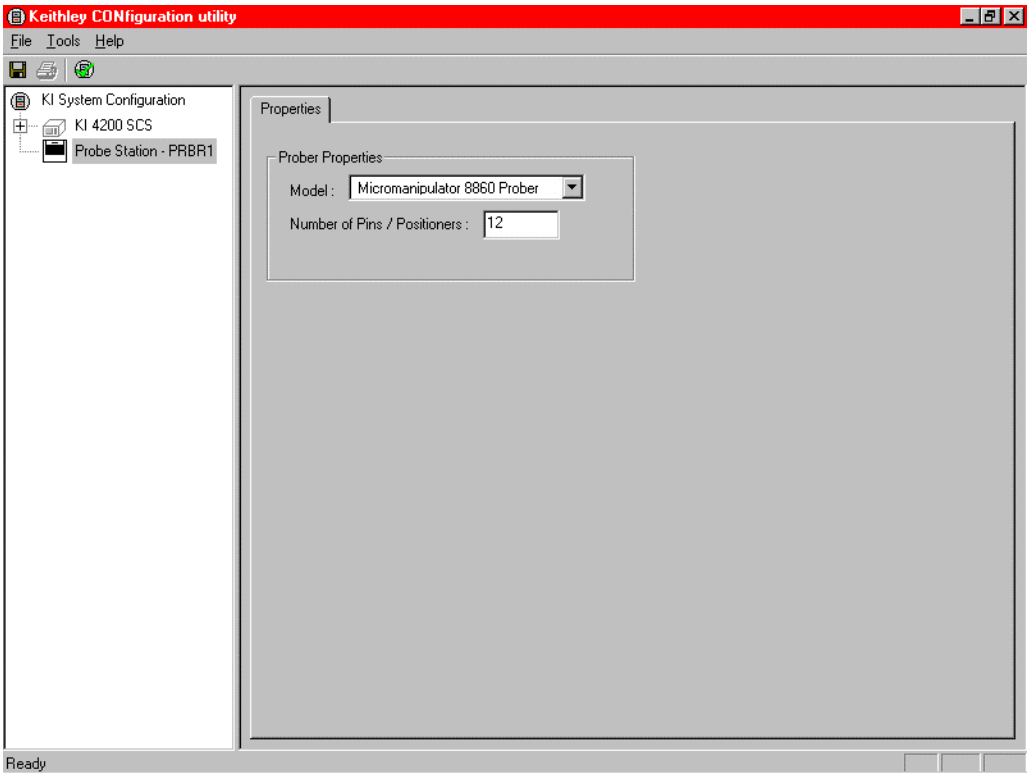
1. From the **Tools** menu (on the Keithley CONfiguration Utility window), click **Add External Instrument Probe Station** (Figure I-44). The probe station **Properties** tab appears.

Figure I-44
KCON: Adding a prober



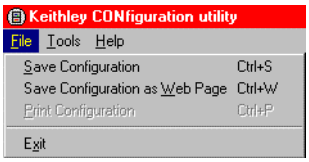
2. Select the Micromanipulator 8860 prober as the model (Figure I-45). Also ensure that the Number of Pins / Positioners is correct.

Figure I-45
KCON: Selecting a prober



3. **Save** the configuration from the **File** menu (Keithley CONfiguration utility window) (Figure I-46).

Figure I-46
KCON: Saving

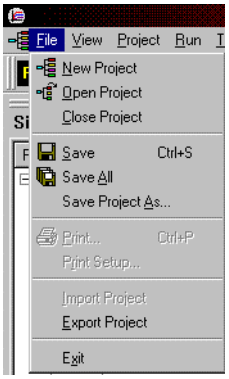


KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

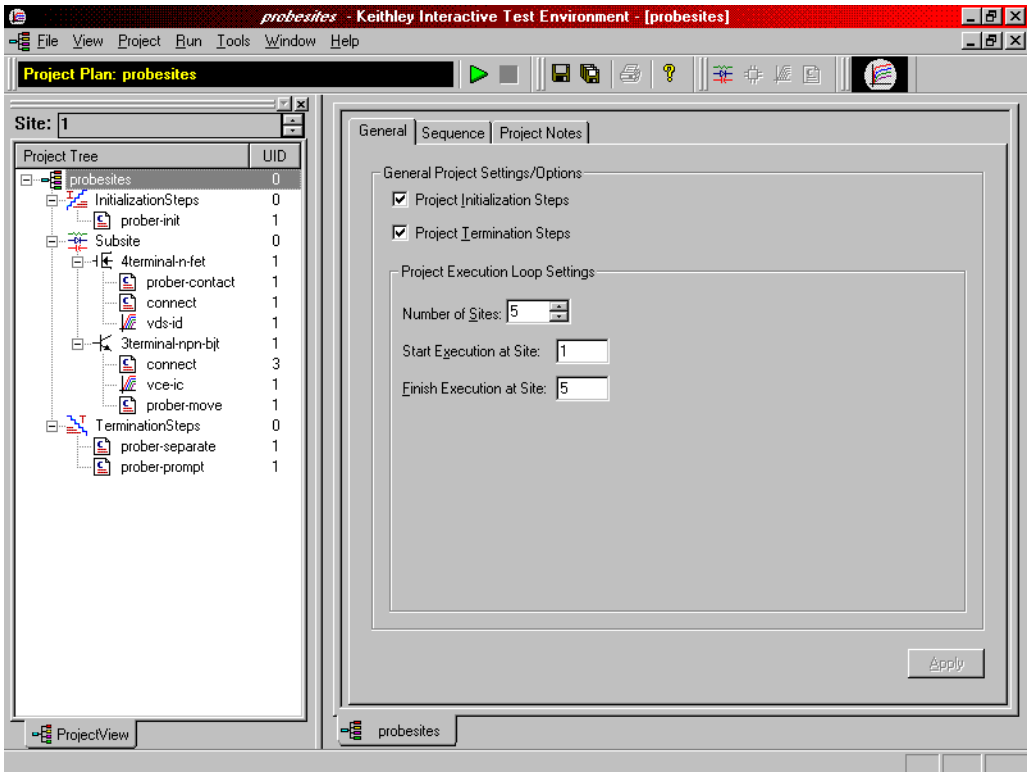
Use KITE to load and run the probesites project:

1. Load the probesites project example from KITE (Figure I-47):
   - Run **KITE**.
   - Select **Open Project** from the **File** menu.



   - **Open** the folder: `c:\S4200\kiuser\Projects\probesites`
   - Select the project file: **probesites.kpr**

Figure I-47
KITE: probesites project



2. Select the top node in the **ProjectView** tab of the Project Navigator window.
3. Click the green **Run** button.

probesubsites KITE project example

The following is a step by step procedure to properly configure the 8860 so the **probesubsites KITE project** executes successfully.

NOTE: *The following configuration is accomplished using the pcBridge computer.*

When using **pcIndie**, ensure that the project and the program listing on the Micromanipulator match (the program listing is a list of absolute chuck moves in the order of execution). When creating the program listing, use a repeatable pattern.

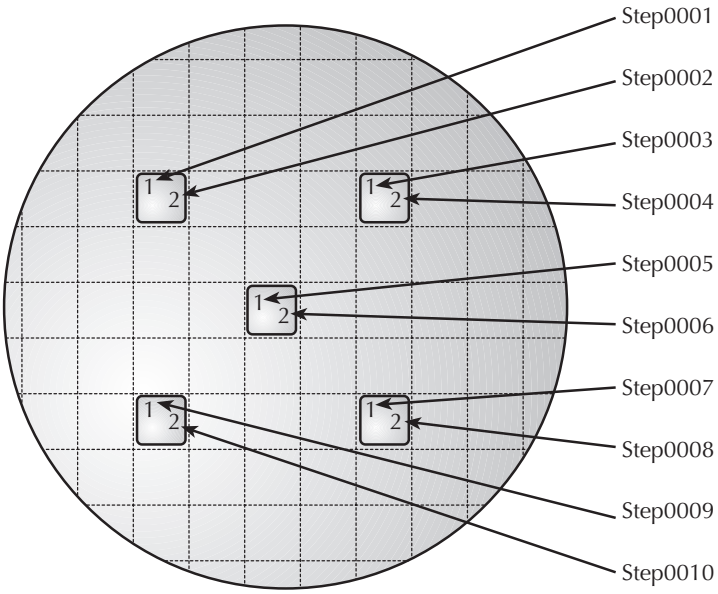
Example Five dies have been selected for probing (Figure I-48). On each die, two sub-sites have been selected.

To configure the 8860:

1) Move to the first subsite of the first die
2) Add it to the program listing
3) Move to the second subsite on the first die
4) Add it to the program listing
5) Move to the first subsite on the second die
6) Add it to the program
7) Continue moving and adding until all subsites have been entered into the list

Using this type of pattern allows the project structure to issue two **PrSSMovNxt** commands within the loop for each die to be probed. Refer to Figure I-48 for an illustration of a repeatable pattern.

Figure I-48
Multiple subsites per die

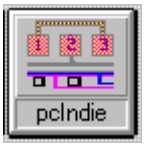


NOTE: *Ensure that all steps of Setup have been completed before starting pcIndie.*

To start pcIndie:

1. Start pcIndie by clicking the **pcIndie** button (Figure I-49) located in the pcLaunch window. The pcIndie window will appear (Figure I-52).

Figure I-49
pcIndie button



2. Using the joystick and microscope, move to the first subsite to be tested.
3. Click **add/ins** into list.

NOTE: *The **add** button adds the description of the present position to the end of the program listing and the **ins** button inserts the present position above the highlighted entry in the program listing. The **rep** button replaces the highlighted entry with the present position and the **del** button deletes the highlighted entry.*

4. Repeat steps 2 and 3 for each subsite to be entered into the program listing.
5. Save by clicking the pcIndie **save** button and assigning the listing a unique file name (*.idp) (Figure I-50).
6. To open an existing program listing file, click the pcIndie **open** button in the pcIndie window. Select the desired file, and click **OK** (Figure I-51).

Figure I-50
pcIndie save button

Figure I-51
pcIndie open button

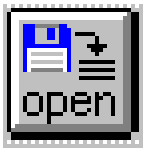
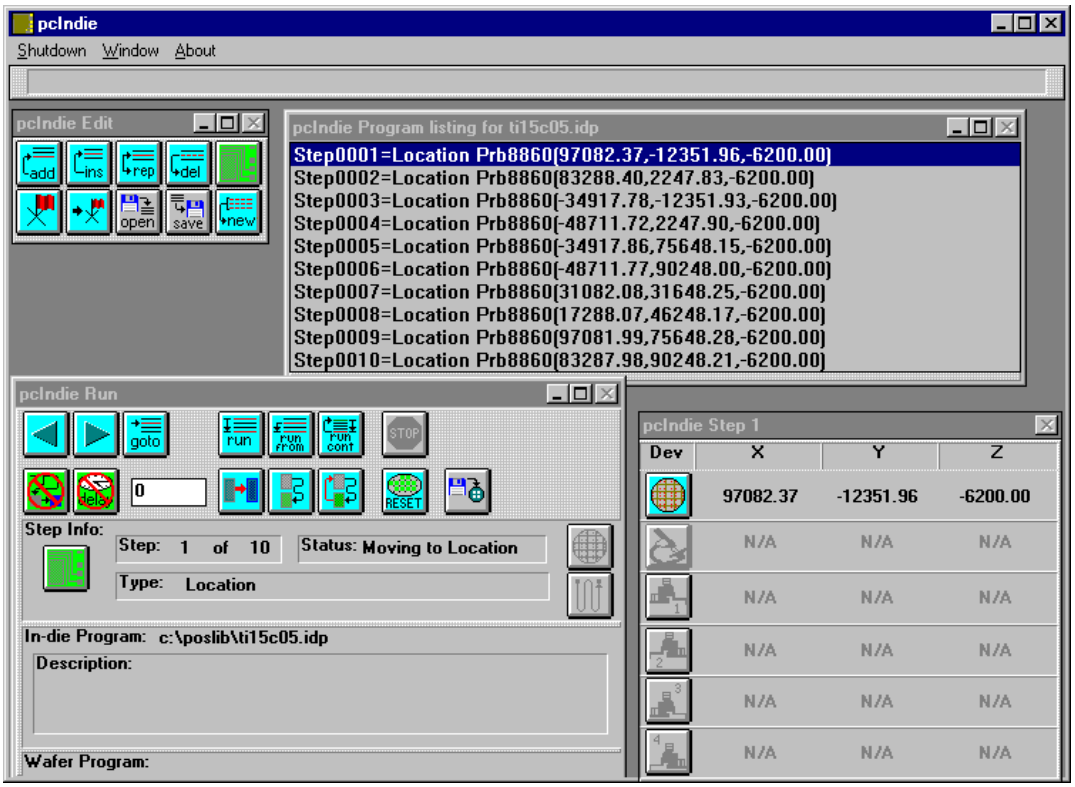


Figure I-52
pcIndie window



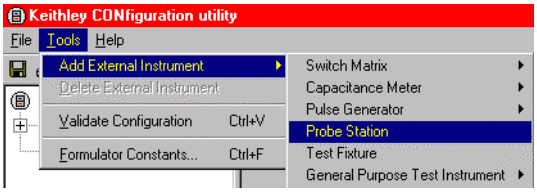
KCON

Use KCON to add the prober to the Model 4200-SCS configuration:

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

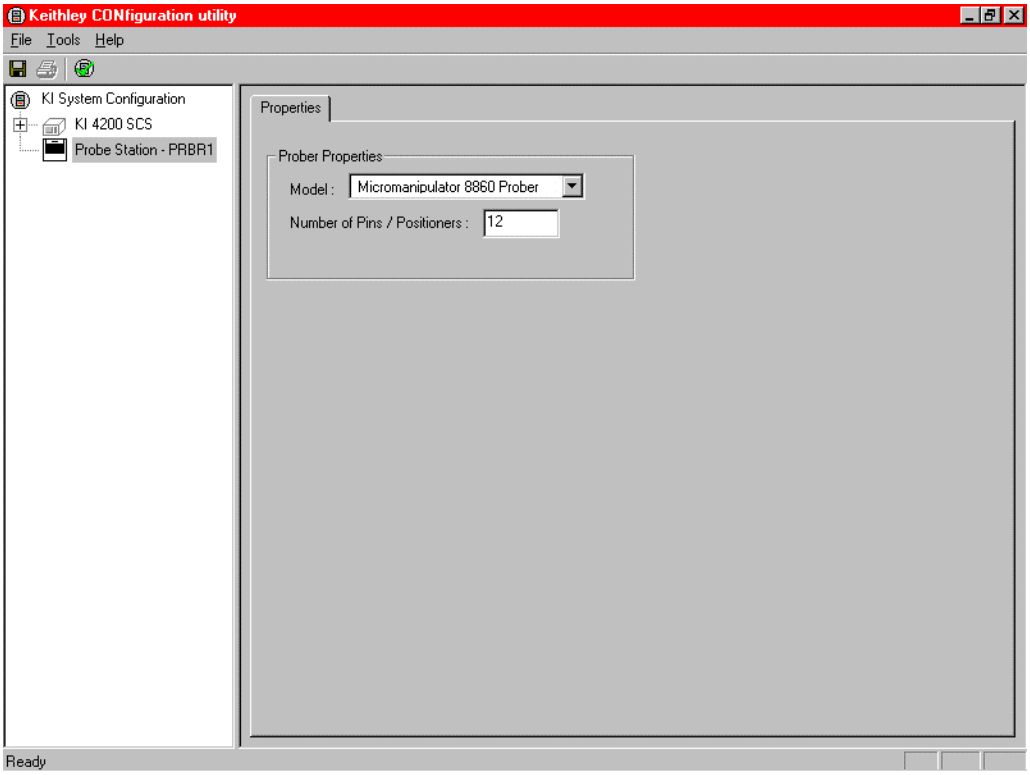
1. From the **Tools** menu (on the Keithley CONfiguration Utility window), click **Add External Instrument > Probe Station** (Figure I-53). The probe station **Properties** tab appears.

Figure I-53
KCON: Adding a prober



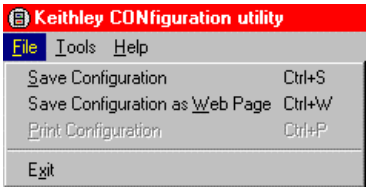
2. Select the Micromanipulator 8860 prober as the model (Figure I-54). Also ensure that the **Number of Pins / Positioners** is correct.

Figure I-54
KCON: Selecting a prober



3. Save the configuration from the **File** menu (Keithley CONfiguration utility window) (Figure I-55).

Figure I-55
KCON: Saving



KITE

Use KITE to load and run the project.

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

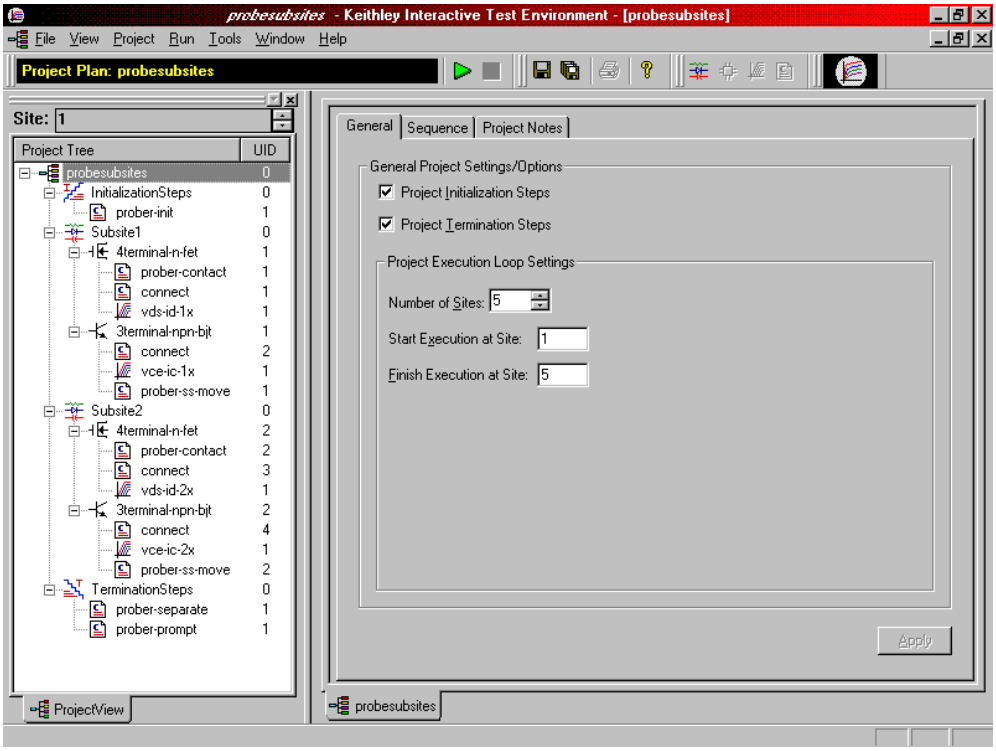
Use the following procedure as a guide to load and run the probesubsites project:

1. Load the probesubsites project example from KITE (Figure I-56).
2. Select the top node in the **ProjectView** tab of the Project Navigator window.
3. Click the green **Run** button.

To load the probesites project:

a. Run **KITE**
b. Select **Open Project** from the **File** menu
c. Open the folder: `c:\S4200\kiuser\Projects\probesubsites`
d. Select the Project File: **probesubsites.kpr.**

Figure I-56
KITE: probesubsites project



Commands and error symbols

The following list (Table I-1) contains error and status symbols listed by command.

Table I-1
Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt
PR_OK	X	X	X	X
BAD_CHUCK	X	X		
UNINTEL_RESP	X	X	X	X
BAD_MODE			X	X
UNEXPE_ERROR		X		
PR_WAFERCOMPLETE			X	X
SET_MODE_FAIL		X		
MOVE_FAIL			X	X

Appendix J
Using a Manual or Fake Prober

In this section:

Required probe station software

The Keithley Instruments Model 4200-SCS provides all software required for both manual or fake prober operation; no additional software is needed.

Probe station configuration

Since remote control (of the prober) is disabled when using manual or fake probers, the probe station must be manually controlled. The user is also responsible for the prober station set-up.

Manual prober overview

Use the MANL prober to test without utilizing automatic prober functionality. Configuring the environment for a MANL prober replaces all computer control of the prober with that of the operator, while allowing the user to step through each command in the sequence. At each prober command, a dialog box will appear instructing the operator what operation is required.

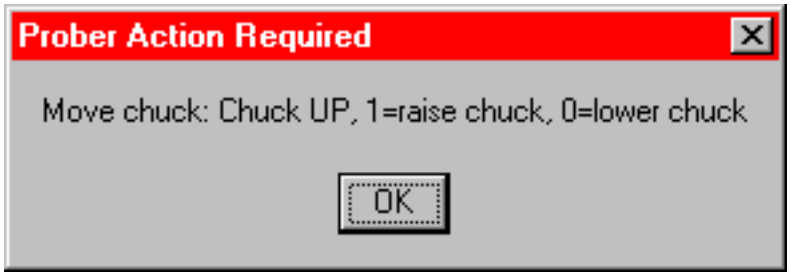
The following describes the probing sequence using the MANL prober:

1. A project is started.
2. A **PrInit** command is issued to tell the user to initialize the prober; the **PrInit** dialog opens.
3. The user continues by selecting **OK.**

Figure J-1
Prober Action Required dialog: OK initialization

4. The project sets up the measurement system to test the first site.
5. A **PrChuck** command is issued to tell the user to ensure the first test site is ready for testing. The **PrChuck** dialog opens (Figure J-1); the user continues by selecting **OK**.

Figure J-2
Prober Action Required dialog: Move chuck



6. Tests on the site are executed.
7. A **PrMovNxt** command is issued to tell the user to move to the next site to be tested on the wafer. The **PrMovNxt** dialog opens (Figure J-13); the user continues by selecting **OK**.

Figure J-3
Prober Action Required dialog: Move probes to next site

8. Steps 5 through 7 are repeated until all sites are tested.

NOTE: *Subsite probing uses the **PrssMovNxt** command in step 6 to move to the next subsite (Figure J-4).*

Figure J-4
Prober Action Required dialog: Move probes to next subsite

Fake prober overview

Use the FAKE prober to test without probing (the prober will not probe). This can be useful to take the prober offline (disable) when you want to run the test without modifying your project. Configuring the environment for a FAKE prober stops all prober actions.

The FAKE prober is useful when prober actions are not desired (for example, when debugging projects). When using the FAKE prober, tests can be executed in a single or looping fashion. This allows debugging of projects without having to remove prober calls. Situations when the FAKE prober mode may be useful:

1. Looping on the same wafer location using a project that supports wafer prober operations (for instance, testing one site 100 times instead of testing 100 different sites once).
2. Disabling prober function calls until the testing portions of the project are functioning correctly.

Modifying the prober configuration file

NOTE: *This file is modified using the Model 4200-SCS computer.*

The default prober configuration file for a manual prober (MANL) is represented in Figure J-5, and for the fake prober, in Figure J-6. As shown, the manual configuration file is configured for use with GPIB prober communications, while the fake configuration file is configured for serial prober communications. To make changes, use Microsoft Windows® Notepad.exe® (no changes are required; the files have been completely configured at the factory).

Configuration file locations: `C:\S4200\sys\dat\prbcnfg_FAKE.dat`
`C:\S4200\sys\dat\prbcnfg_MANL.dat`

Figure J-5
Sample manual prober configuration file

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File MANL Prober
#
# The following tag, "PRBCNFG", is used by the engine in order to
determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#       01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#  OcrPresent
#  AutoAlnPresent
#  ProfilerPresent
#  HotchuckPresent
#  HandlerPresent
#  Probe2PadPresent
#
# Configuration for MANuaL probers (S900NT):
#  MANL
#
PROBER_1_PROBTYPE=MANL
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=5
PROBER_1_GPIB_WRITEMODE=0
PROBER_1_GPIB_READMODE=2
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
```

NOTE: *The highlighted line in the configuration file (Figure J-5) is the only relevant line for this prober type.*

Figure J-6
Sample fake prober configuration file

```
# prbcnfg.dat  - EXAMPLE Prober Configuration File, FAKE prober
#
# The following tag, "PRBCNFG", is used by the engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#          01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#  OcrPresent
#  AutoAlnPresent
#  ProfilerPresent
#  HotchuckPresent
#  HandlerPresent
#  Probe2PadPresent
#
#
# The PROBER_x_PROBTYPE fields needs to be set to one of the following names.
# Configuration for serial probers:
#  FAKE
#
#
PROBER_1_PROBTYPE=FAKE
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=SERIAL
PROBER_1_DEVICE_NAME=COM1
PROBER_1_BAUDRATE=9600
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
```

NOTE: *The highlighted line in the configuration file (Figure J-6) is the only relevant line for this prober type.*

Probesites KITE project example

The following is a step-by-step procedure to properly configure the manual prober so the **probesites** Keithley Interactive Test Environment (KITE) project executes successfully. The user is responsible for the probe station set-up.

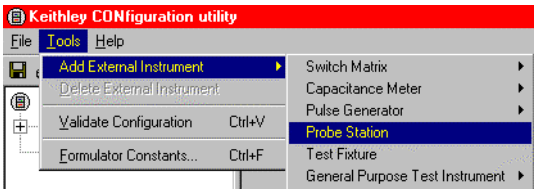
Keithley CONfiguration (KCON) utility

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer. Use the following procedure as a guide to add a prober to the Model 4200-SCS.*

Use KCON to add the prober to the configuration.

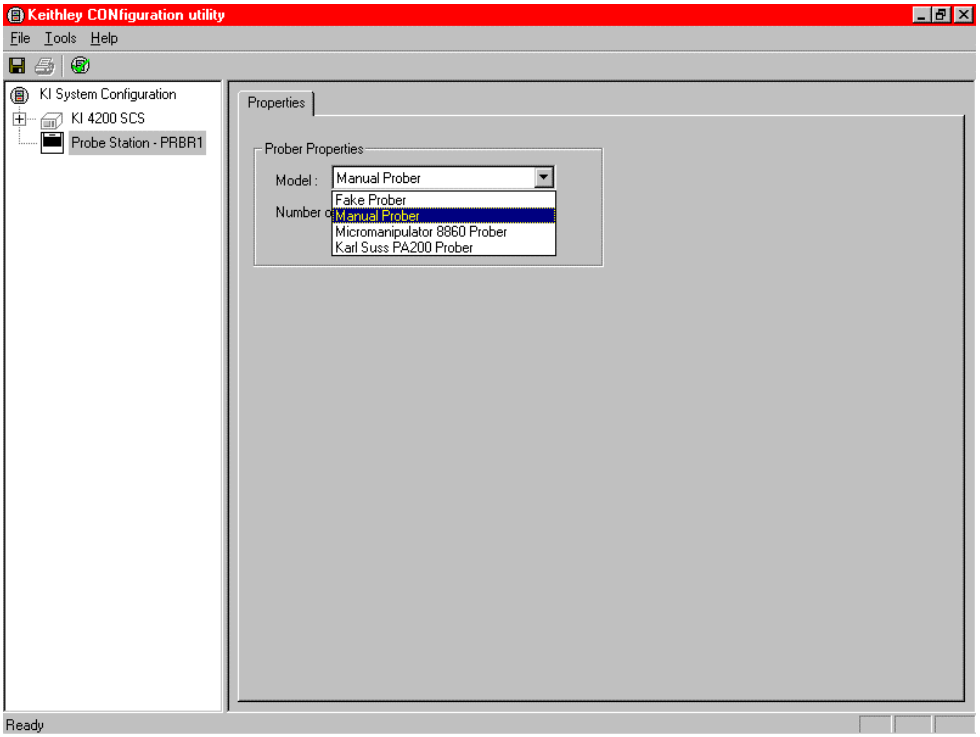
1. From the **Tools** menu on the Keithley CONfiguration utility window, click **Add External Instrument > Probe Station** (Figure J-7). The probe station **Properties** tab appears (Figure J-8).

Figure J-7
KCON: Adding a prober



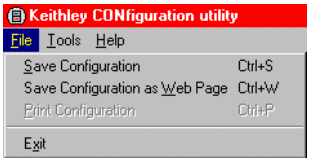
2. Select the **Manual Prober** or the **Fake Prober** as the model.

Figure J-8
KCON: Selecting a prober



3. Save the configuration from the **File** menu on the Keithley CONfiguration utility window (Figure J-9).

Figure J-9
KCON: Saving



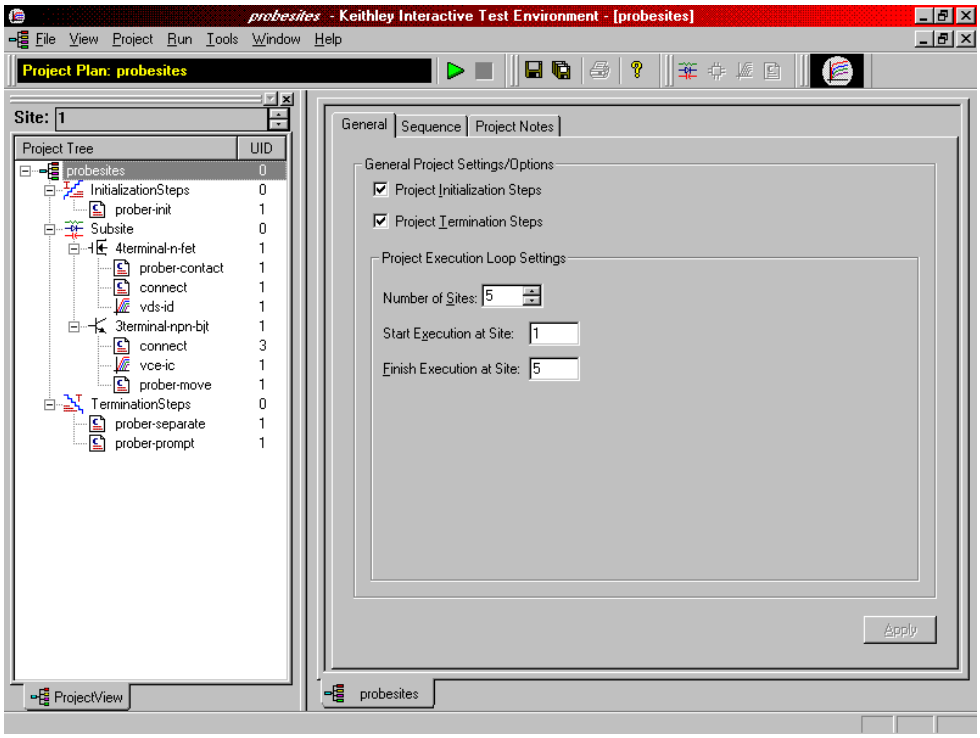
Keithley Interactive Test Environment (KITE)

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KITE to load and run the **probesites project** using the new configuration file which will allow you to execute the project for this prober.

1. Load the **probesites project** example from KITE (Figure J-10):
   a. Run **KITE**.
   b. Select **Open Project** from the **File** menu.
   c. Open the folder: `c:\S4200\kiuser\Projects\probesites`
   d. Select the project file: **probesites.kpr**
2. Select the top node in the **ProjectView** tab of the Project Navigator window (Figure J-10).

Figure J-10
KITE: probesites project



3. Click the green **Run** button.

Probesubsites KITE project example

The following is a step-by-step procedure to properly configure the Manual Prober so the **probesubsites KITE project** executes successfully. The user is responsible for the probe station set-up.

Keithley CONfiguration (KCON) utility

Use KCON to add the prober to the configuration:

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

1. From the **Tools** menu on the Keithley CONfiguration utility window, click **Add External Instrument > Probe Station** (Figure J-11). The probe station **Properties** tab appears (Figure J-12).

Figure J-11
KCON: Adding a prober

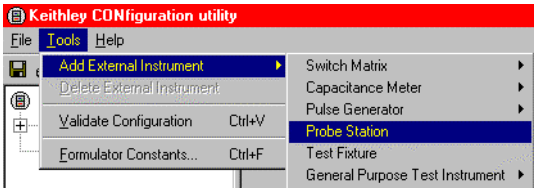
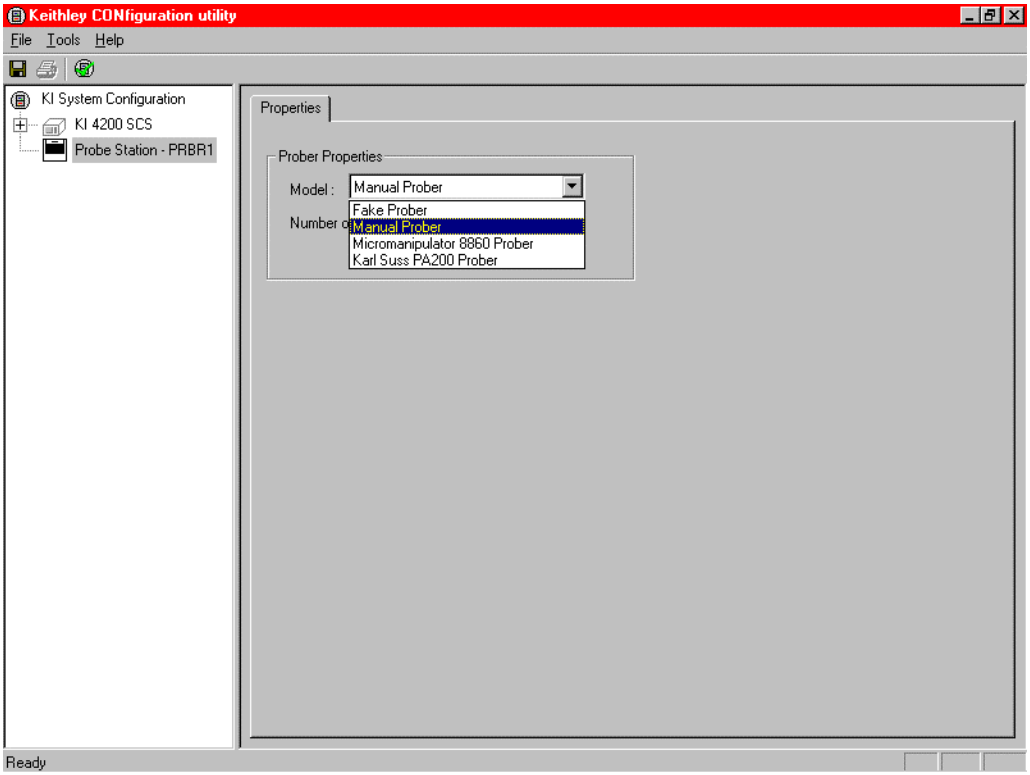


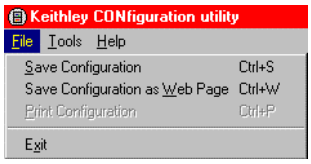
Figure J-12
KCON: Selecting a prober



2. Select the **Manual Prober** or the **Fake Prober** as the model.

3. Save the configuration from the **File** menu (on the Keithley CONfiguration utility window) (Figure J-13).

Figure J-13
KCON: Saving



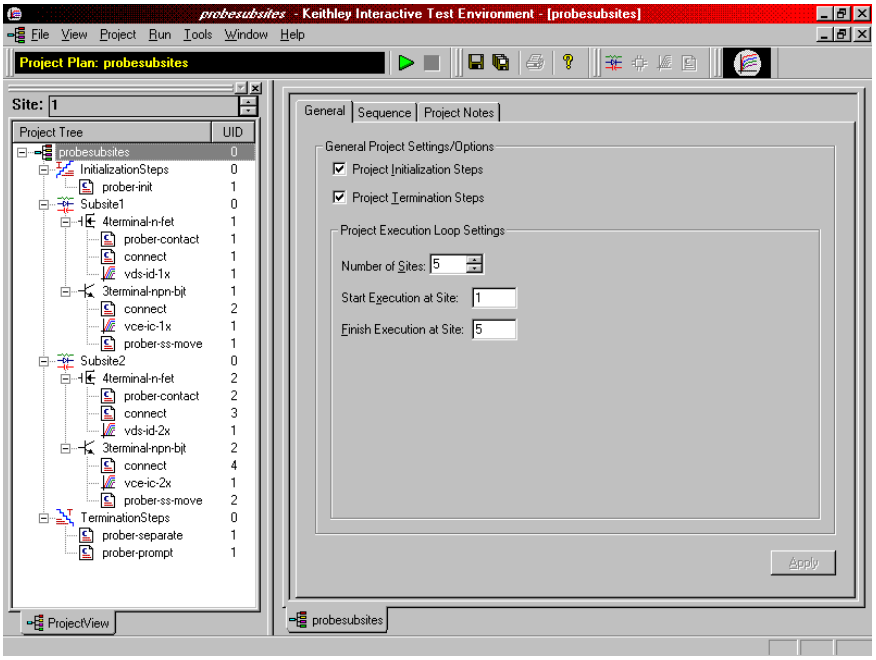
Keithley Interactive Test Environment (KITE)

Use KITE to load and run the **probesubsites** project using the new configuration file, which will allow you to execute the project for this prober.

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

1. Load the **probesubsites** project example from KITE (Figure J-14):
   a. Run **KITE.**
   b. Select **Open Project** from the **File** menu.
   c. Open the folder: `c:\S4200\kiuser\Projects\probesites.`
   d. Select the project file: **probesites.kpr.**

Figure J-14
KITE: probesubsites project



2. Select the top node in the ProjectView tab of the Project Navigator window.
3. Click the green **Run** button.

Appendix K
Cascade Summit-12000 Prober

In this section:

Required probe station software

The following program is used to configure and operate the SUMMIT-12000 prober with the Keithley Instruments Model 4200-SCS:

Nucleus UI: Provides easy access to configuration and help programs.

Software versions

The following software version was used to verify the configuration of the SUMMIT-12000 prober with the Model 4200-SCS:

Nucleus UI ver. 2.0

Probe station configuration

CAUTION **Although this appendix provides instructions on prober setup and configuration, make sure that you are familiar with the Cascade SUMMIT-12000 Prober and its supporting documentation before attempting setup, configuration, or operation.**

There are four general steps required to setup and configure the PA-200 prober for use with the Model 4200-SCS:

- Step 1. Set up communication
- Step 2. Set up wafer geometry
- Step 3. Create a site definition and define a probe list
- Step 4. Load, align, and contact the wafer

Each step is detailed later in this section.

Modifying the prober configuration file

NOTE: *This file is modified using the Model 4200-SCS computer.*

The default prober configuration file is contained in Figure K-1. As shown, the file is configured for use with a GPIB communication setup. Use Notepad.exe to open, modify, and save this file should any change be required.

Configuration file location: `C:\S4200\sys\dat\prbcnfg_CC12K.dat.`

Figure K-1
Sample SUMMIT-12K prober configuration file

```
# prbcnfg_CC12K.dat - DEFAULT Prober Configuration
File
#
# The following tag, "PRBCNFG", is used by the
engine in order to determine
# the MAX number of SLOTS and CASSETTES for a given
prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
# 01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#OcrPresent
#AutoAlnPresent
#ProfilerPresent
#HotchuckPresent
#HandlerPresent
#Probe2PadPresent
#
#
# Configuration for direct GPIB probers:
# CC12K
#
PROBER_1_PROBTYPE=CC12K
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=28
PROBER_1_GPIB_WRITEMODE=0
PROBER_1_GPIB_READMODE=2
PROBER_1_GPIB_TERMINATOR=13
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
```

Step 1. Set up communication

The Cascade SUMMIT-12000 prober is configured for GPIB communication only. Ensure that the prober configuration file is set up properly for the GPIB communication interface (see Modifying the prober configuration file earlier in this appendix).

NOTE: *The following configuration is accomplished using the probe station PC.*

1. Connect the Model 4200-SCS GPIB port and the probe station PC's GPIB port using a shielded IEEE-488 cable (Model 7007). Refer to Figure K-2 and Table K-1 for pinouts, and to Figure K-3 for a connection diagram.

Figure K-2
IEEE-488 connector

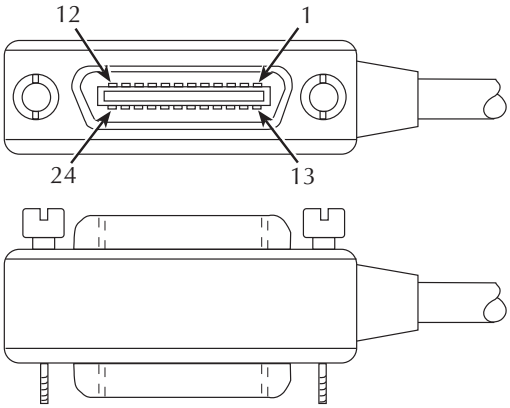
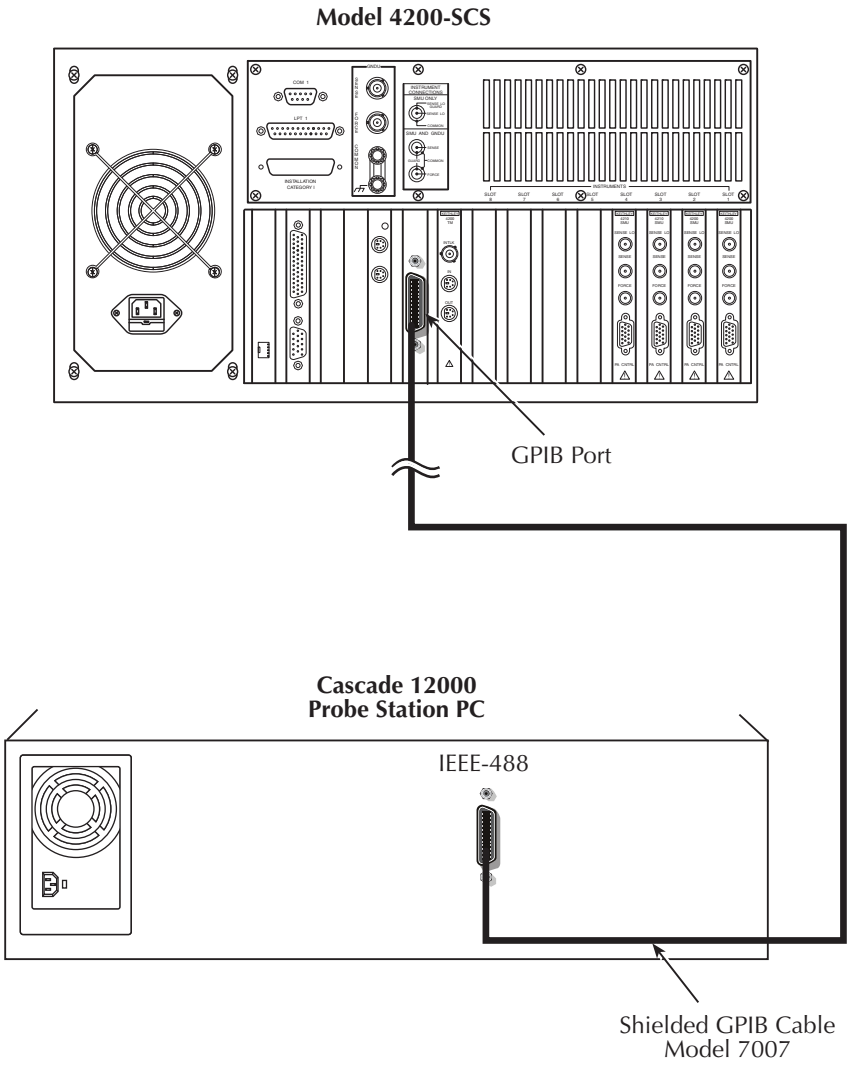


Table K-1
IEEE-488 control connector terminals

Contact number	IEEE-488 designation	Type
1	DI01	Data
2	DI02	Data
3	DI03	Data
4	DI04	Data
5	EOI (24)*	Management
6	DAV	Handshake
7	NRFD	Handshake
8	NDAC	Handshake
9	IFC	Management
10	SRQ	Management
11	ATN	Management
12	SHIELD	Ground
13	DI05	Data
14	DI06	Data
15	DI07	Data
16	DI08	Data
17	REN (24)*	Management
18	Gnd (6) *	Ground
19	Gnd (7) *	Ground
20	Gnd (8) *	Ground
21	Gnd (9) *	Ground
22	Gnd (10) *	Ground
23	Gnd (11) *	Ground
24	Gnd, LOGIC	Ground

* Numbers in parentheses refer to signal ground return of referenced contact number. EOI and REN signal lines return on contact 24.

Figure K-3
Connection diagram

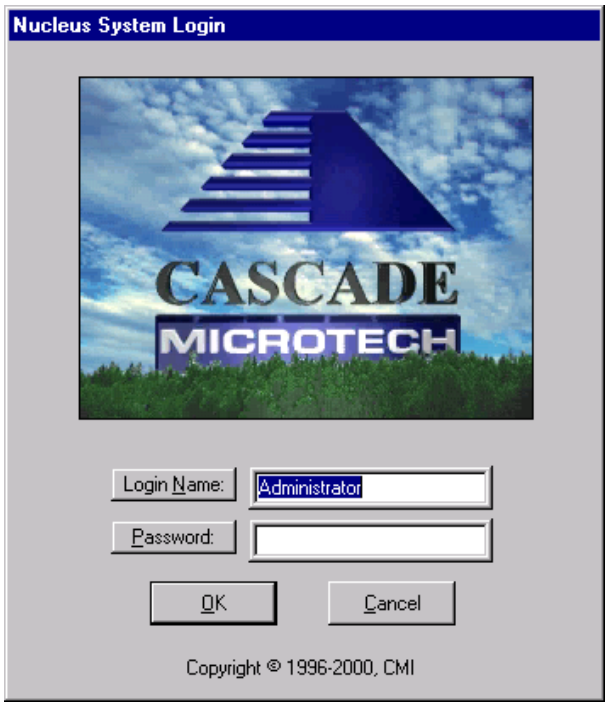


2. Double-click the **Nucleus** icon on the Microsoft® Windows® desktop (Figure K-4).

Figure K-4
Nucleus icon

3. Login using the **Nucleus System Login** (Figure K-5).
4. After login is complete, the prober will initialize the stage. Click **proceed** when the prober has completed initialization.

Figure K-5
Login window

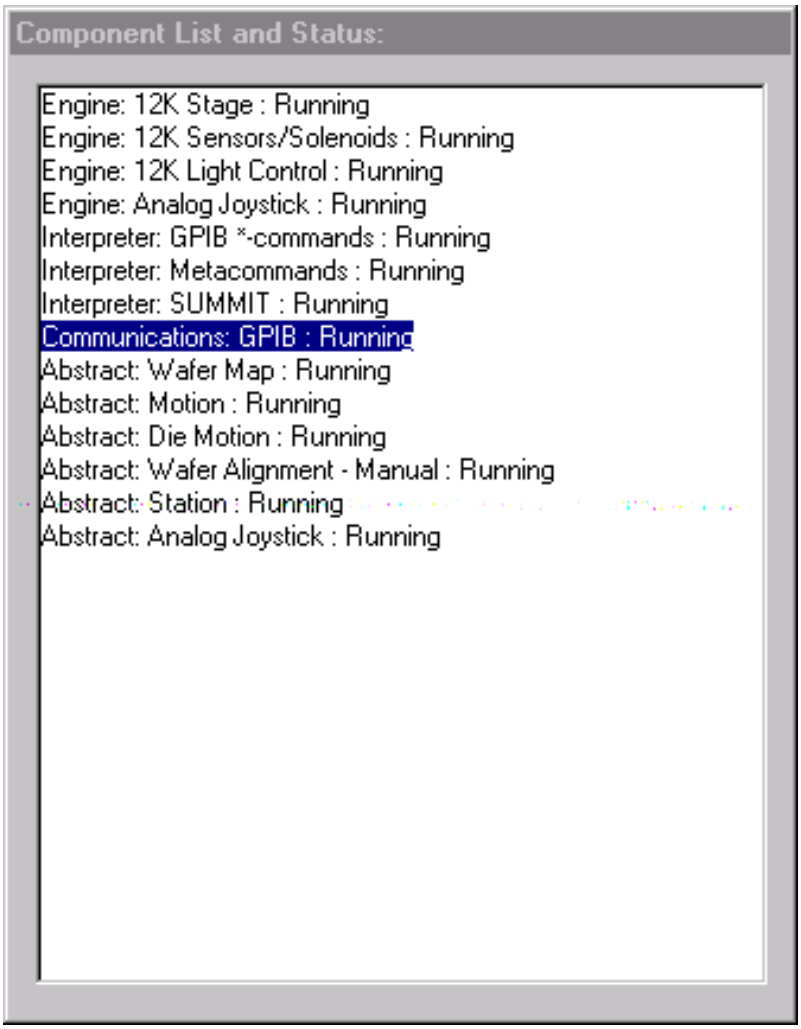


5. Maximize the system managers Component List and Status program (right-click the **system manager** label on the taskbar and choose **Maximize**).
6. Select (click) the **Communications: GPIB** component on the component list (Figure K-6).

NOTE: *If the* **Communications: GPIB** *component is not on the list, then it must be added.*

To add: Click **Add** *from the Add component dialog, then select* ***Communications: GPIB.***

Figure K-6
Component list and status window



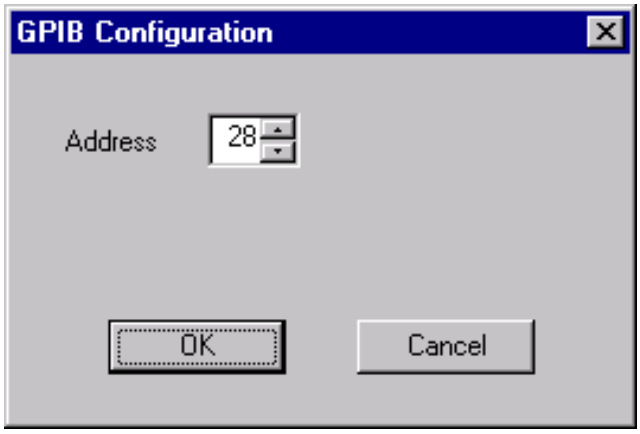
7. If the **Communications: GPIB** component is running, click the **Stop** button (Figure K-7)**,** or else proceed to the next step (setup).

Figure K-7
Stop button

8. Click the **Setup** button (Figure K-8) to open the GPIB configuration dialog (Figure K-9).

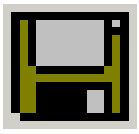
Figure K-8
Setup button

Figure K-9
GPIB Configuration window



9. Change the address as desired. The default value is 28.
10. Save the configuration file by clicking the **Save** button (Figure K-10).

Figure K-10
Save button



11. Start the component by clicking the **GO** button.

Figure K-11
GO button

12. Minimize, but do not close, the system manager window.
13. Click the **Remote** button (Figure K-12) on the Nucleus UI toolbar (Figure K-13) to display the Remote window (Figure K-14).

Figure K-12
Remote button

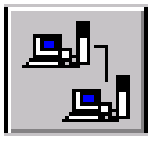


Figure K-13
Nucleus UI toolbar

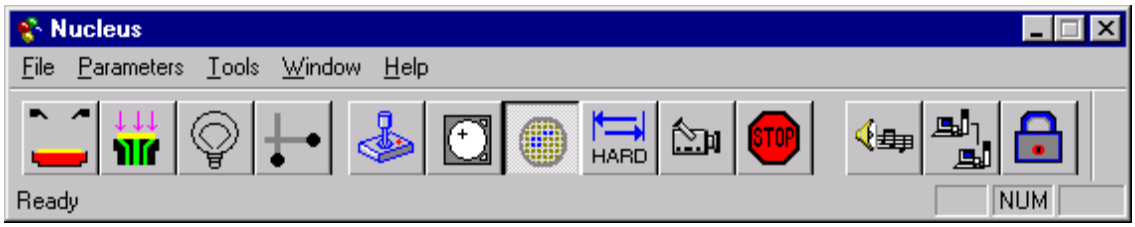
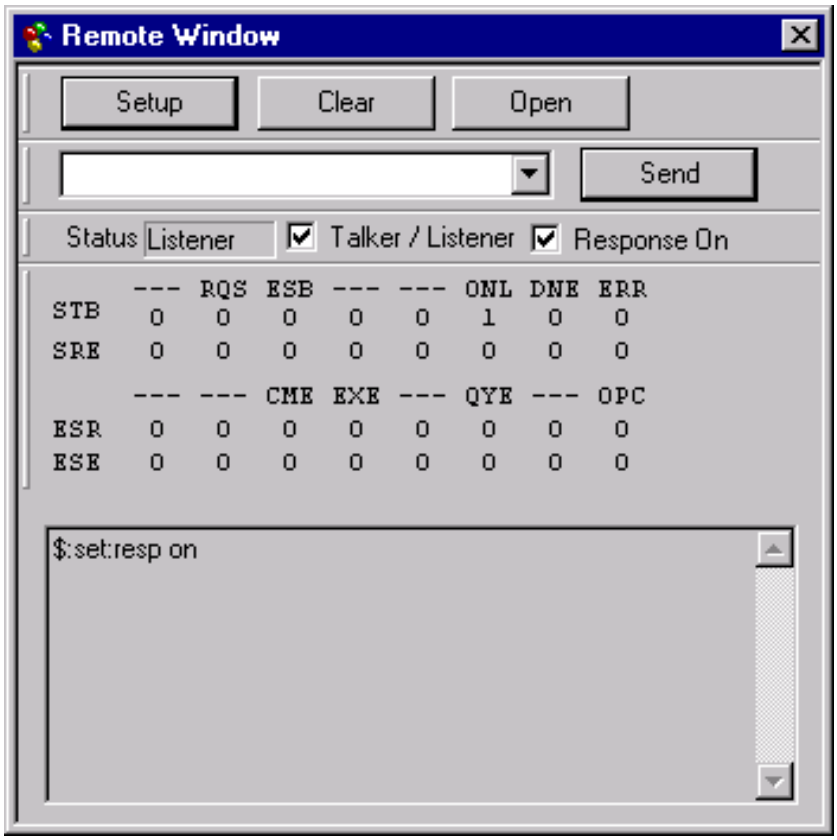


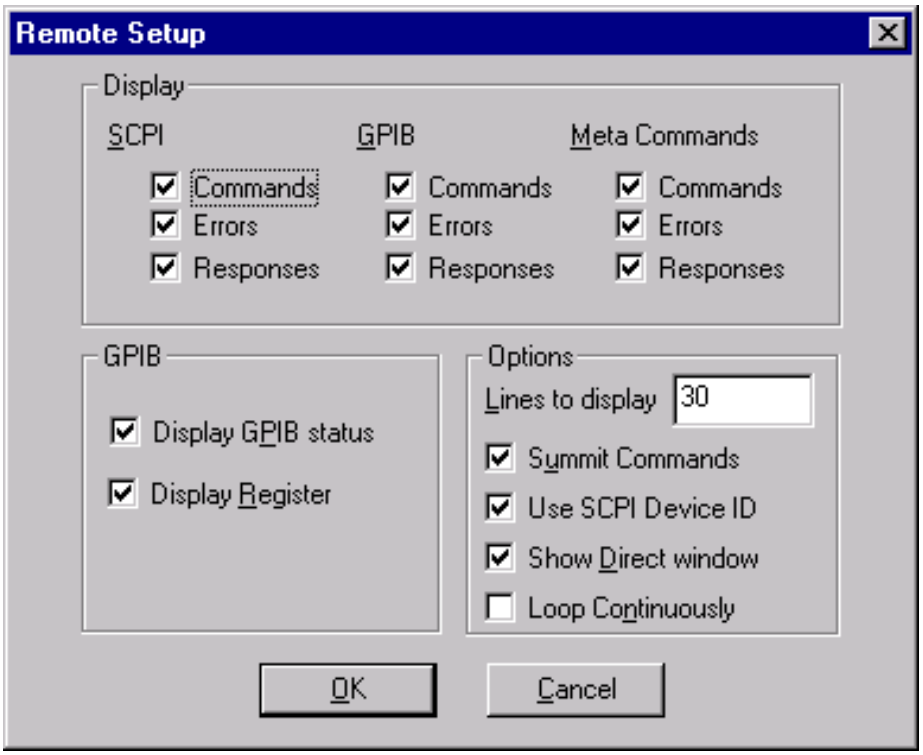
Figure K-14
Remote Window



14. Check the **Talker / Listener** and **Response On** boxes contained in the Remote Window (Figure K-14).
15. Click **Setup** button on the Remote Window (Figure K-14) to display the Setup Window (Figure K-15).
16. Check the desired items to be displayed.
17. Click **OK**.

NOTE: *Checking boxes on the setup window only affects the DISPLAY properties. It will not change the GPIB physical setting. Use the dialog box contained in Figure K-9 to make changes to the GPIB address.*

Figure K-15
Remote setup window



Step 2. Set up wafer geometry

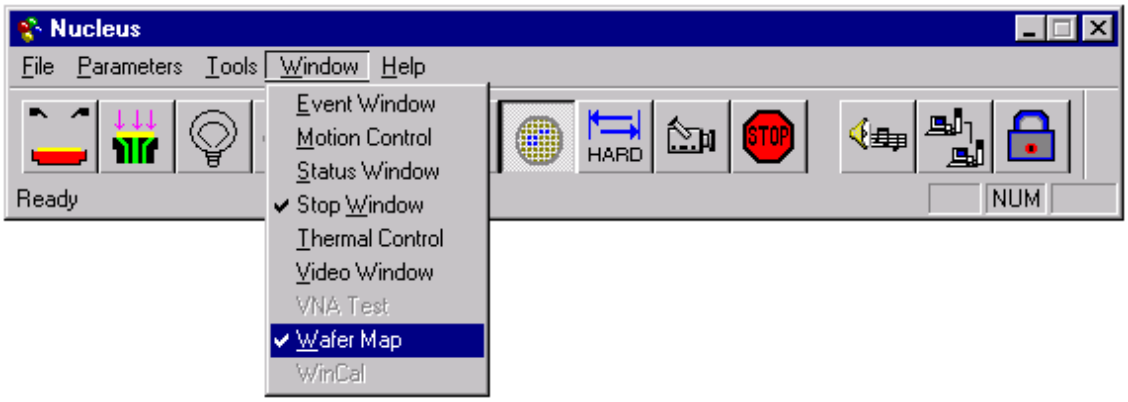
NOTE: *The following configuration is accomplished using Nucleus UI on the probe station PC.*

1. If the Nucleus toolbar (Figure K-17) is not already open, double-click the **Nucleus** icon on the Windows desktop (Figure K-16).

Figure K-16
Nucleus icon

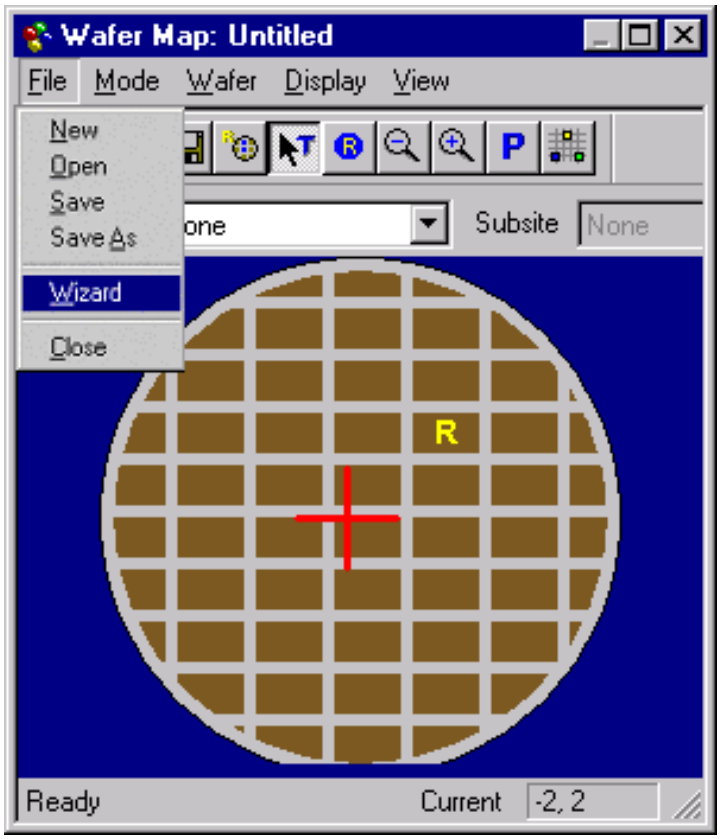
2. Log in.
3. From the Window menu of Nucleus toolbar (Figure K-17), select **WaferMap** to display the wafer map window (Figure K-18).

Figure K-17
Nucleus toolbar



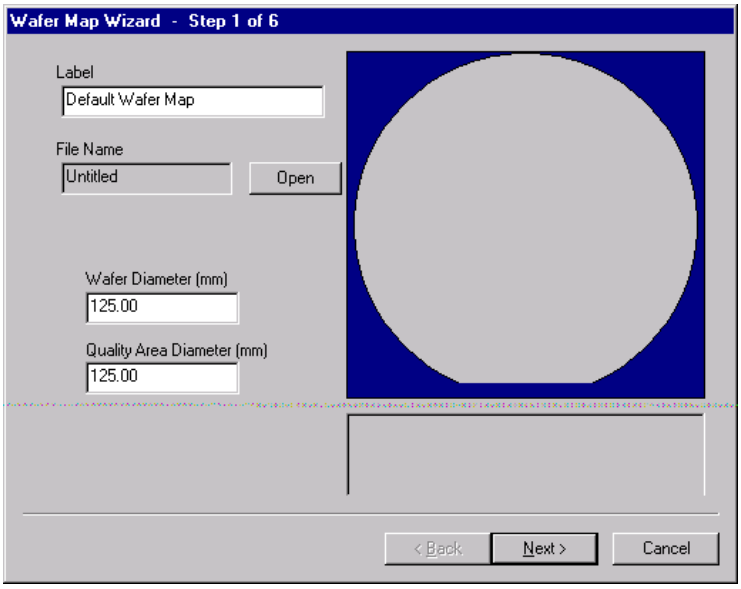
4. From the **File** menu of the Wafer Map window, select **Wizard** to start the Wafer Map wizard (Figure K-19).

Figure K-18
Wafer Map window



5. Enter the label and wafer diameter in the Wafer Map Wizard window (Figure K-19).

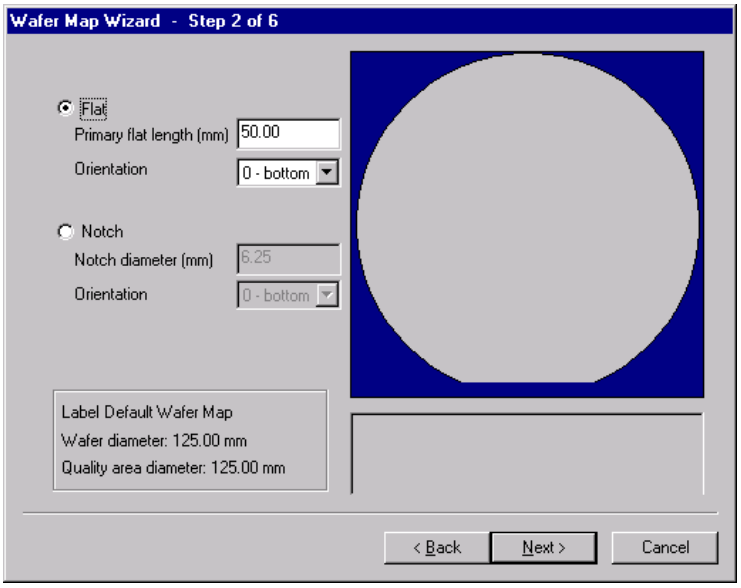
Figure K-19
Step 1: Wafer Map Wizard



6. Click **Next**.
7. Select **Flat** or **Notch** based on the actual wafer.
8. Enter either the primary flat length or the notch diameter in millimeters.
9. Select the orientation of the flat or notch as applicable (Figure K-20).

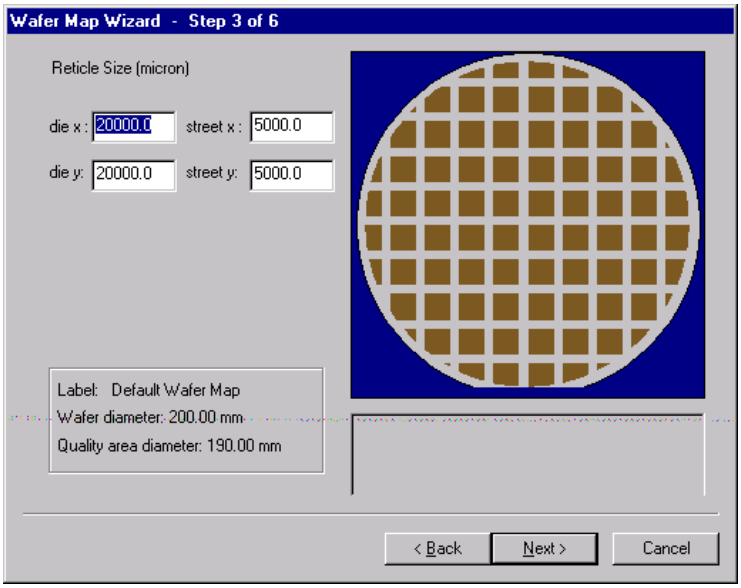
NOTE: *Bottom is toward front of prober.*

Figure K-20
Step 2: Wafer Map Wizard



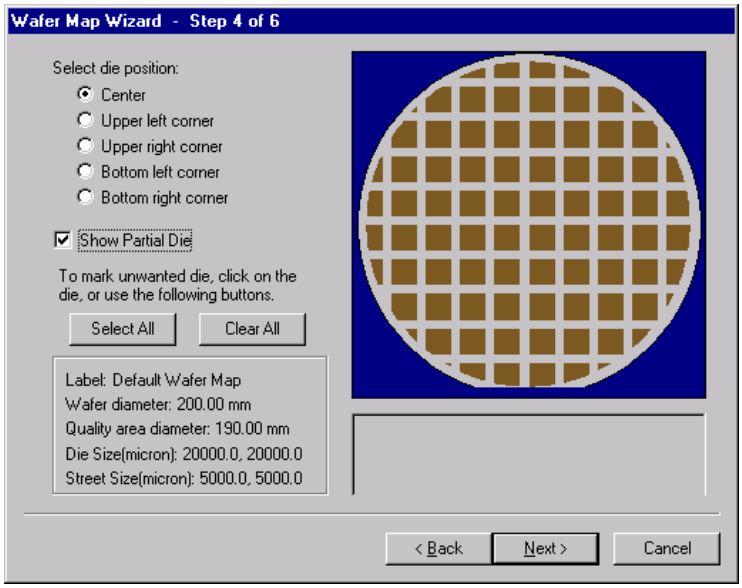
10. Click **Next**.
11. Enter the correct die and street sizes (Figure K-21).

Figure K-21
Step 3: Wafer Map Wizard



12. Click **Next**.
13. Select the die position. Optionally, check the **Show Partial Die** box (Figure K-22).

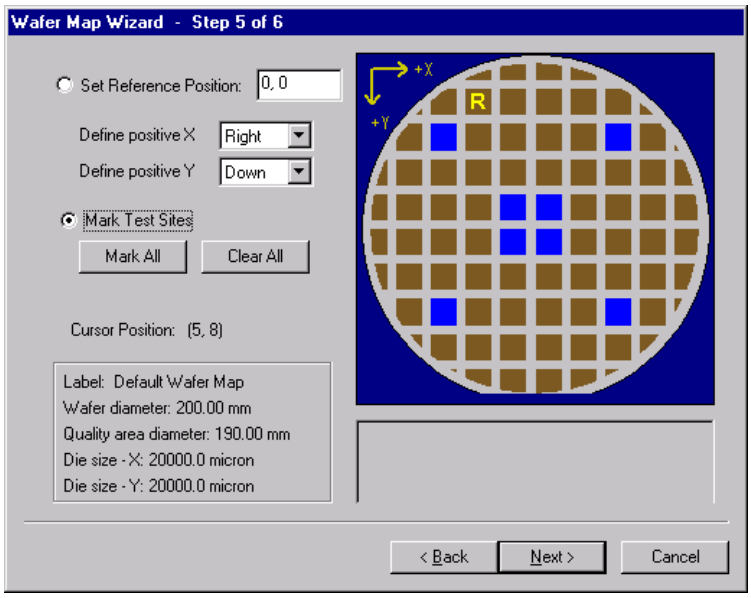
Figure K-22
Step 4: Wafer Map Wizard



14. Click **Next**.
15. Set the reference position (Figure K-23).
16. Enter positive X and Y value directions (this defines the coordinate). For example, setting **Define Positive X: Right**, and **Define Positive Y: Up** would define the coordinate as Quadrant I, while setting **Define Positive X: Right**, and **Define Positive Y: Down** would define the coordinate as Quadrant IV.
17. Click **Mark Test Sites**.

NOTE: *Click and drag to select multiple sites.*

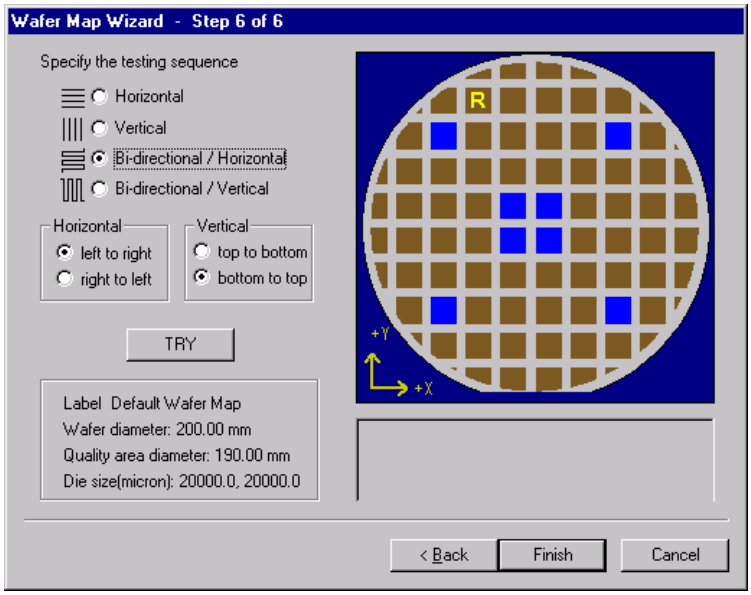
Figure K-23
Step 5: Wafer Map Wizard



NOTE: *Refer to the probesites and probesubsites KITE project examples for specifics on selecting sites to probe.*

18. Click **Next**.
19. Specify the test sequence (Figure K-24).

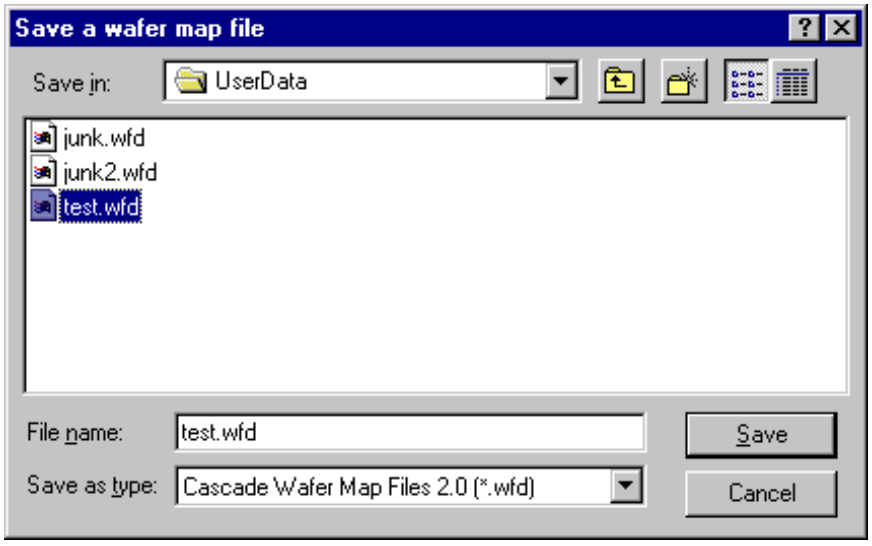
Figure K-24
Step 6: Wafer Map Wizard



20. Click **Finish**.

21. Save the **Wafer Map** settings (Figure K-25).

Figure K-25
Save Wafer Map



Step 3. Create a site definition and define a probe list

NOTE: *The following setup procedure is accomplished using Nucleus UI on probe station PC.*

Creating a site definition for single subsite per die involves using the software to create a selection of dies to probe. If a single subsite per die is to be probed, refer to Probesites KITE Project example later in this appendix. Creating a site definition for multiple subsites per die also involves using the software to create a selection of dies to probe, but also includes creating a selection of the subsites on each die that will be probed. If multiple subsites per die will be probed, refer to Probesubsites KITE Project example later in this appendix.

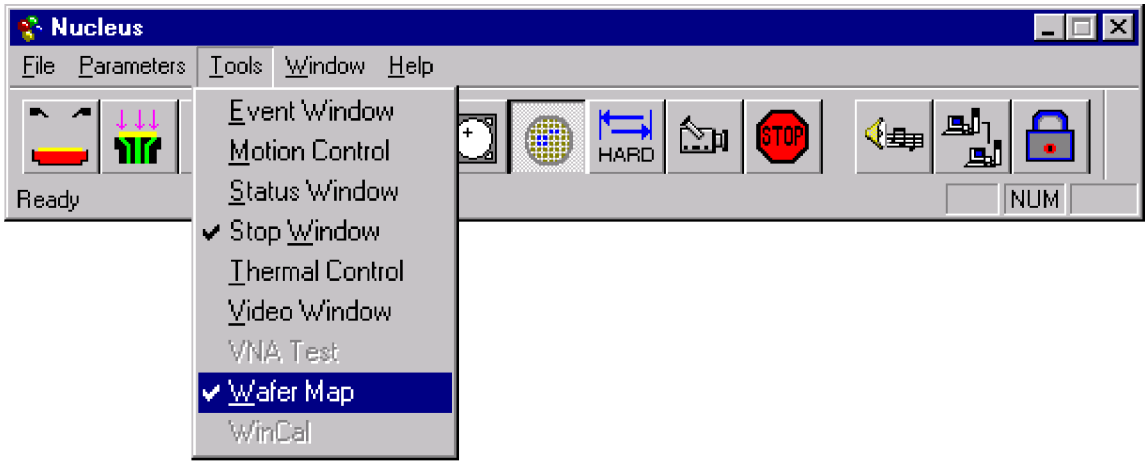
To open a previously defined and saved site definition and a probe list:

1. If the **Nucleus** toolbar is not already open, double-click the **Nucleus UI** icon on the Windows desktop (Figure K-26).

Figure K-26
Nucleus UI icon

2. Log in.
3. From the **Nucleus** toolbar (Figure K-27), select **Tools > WaferMap** (Figure K-27). The Wafer Map window will be displayed (Figure K-28).

Figure K-27
Nucleus toolbar



4. From the wafer Map Window, select **File > Open** (Figure K-28).
5. Open the desired **wafer map** file (Figure K-29).

Figure K-28
Wafer Map window

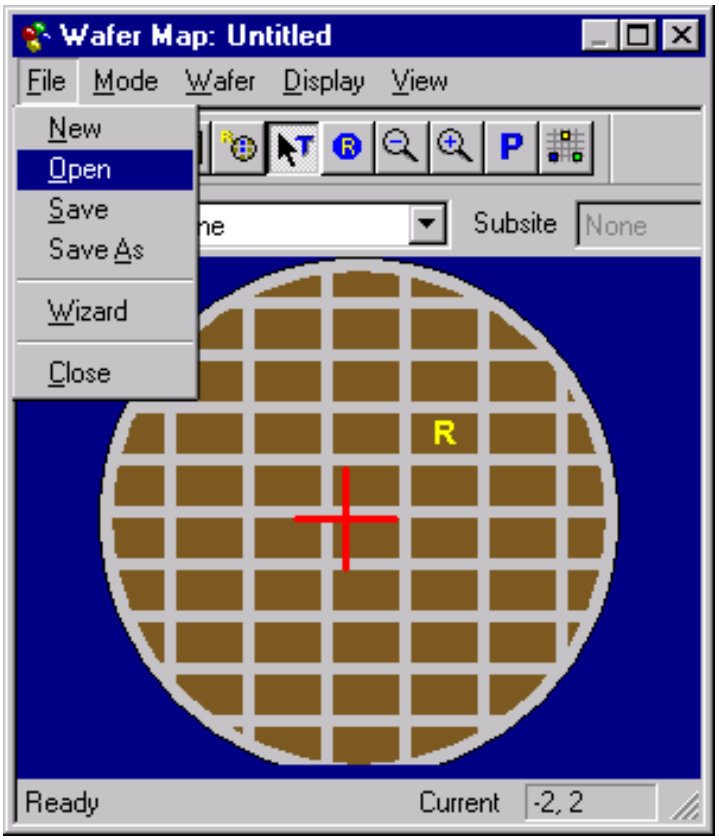
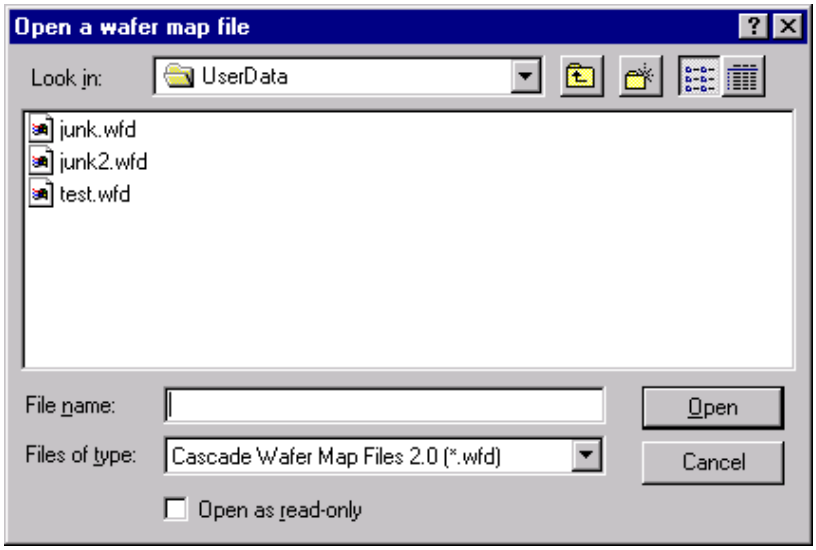


Figure K-29
Open a wafer map file window



Step 4. Load, align, and contact the wafer

NOTE: *The following procedure is accomplished using Nucleus UI on the probe station PC.*

1. From the **Nucleus** toolbar (Figure K-30), select **Window** → **Motion Control**. The Motion Control window will open (Figure K-31).

Figure K-30
Nucleus toolbar

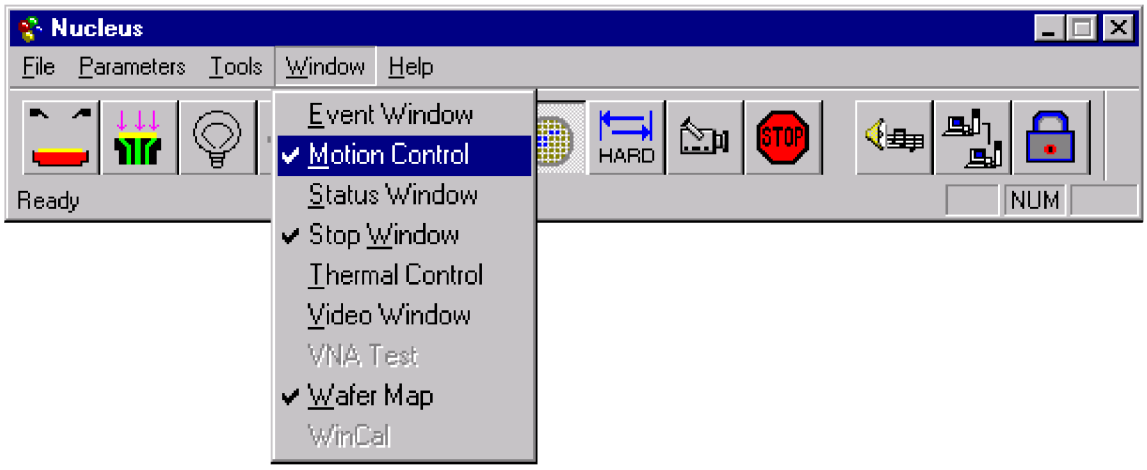
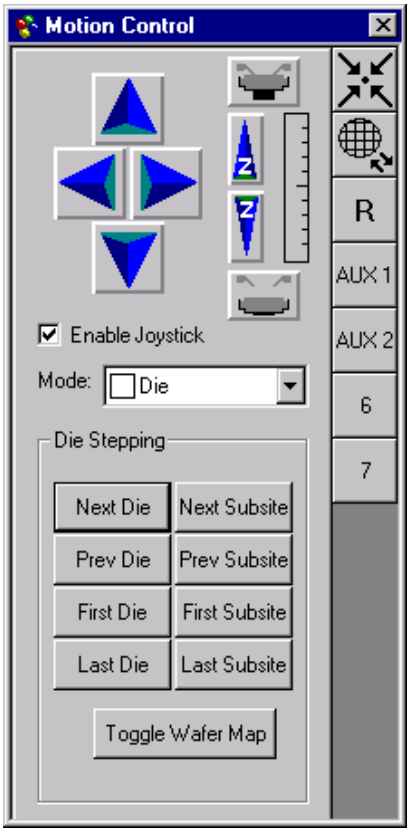
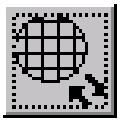


Figure K-31
Motion Control window



2. From the **Motion Control** window, click the **Chuck to front** button (Figure K-32).

Figure K-32
Chuck to front button



3. From the **Nucleus** toolbar (Figure K-30), click the **Enable Joystick** button (Figure K-33).

Figure K-33
Enable Joystick button

4. Place a wafer on chuck.
5. From the Nucleus UI toolbar, toggle the vacuum from **OFF** to **ON** (Figure K-34).

Figure K-34
Vacuum control

6. From the Nucleus UI toolbar, turn on the camera screen by clicking the **Video** button (Figure K-35).

Figure K-35
VIDEO

NOTE: *If the LIGHT is off, the video will be blank.*

7. From the Nucleus UI toolbar, turn on the light by clicking the **LIGHT** button (Figure K-36).

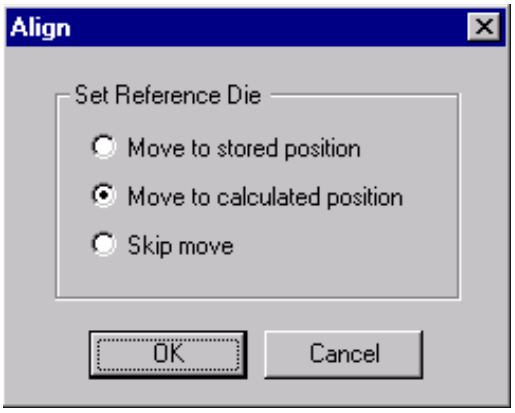
Figure K-36
Light button

8. From the **Wafer Map** window, click the **Reference Die** button (Figure K-37). The Align dialog box will open (Figure K-38).

Figure K-37
Reference Die button

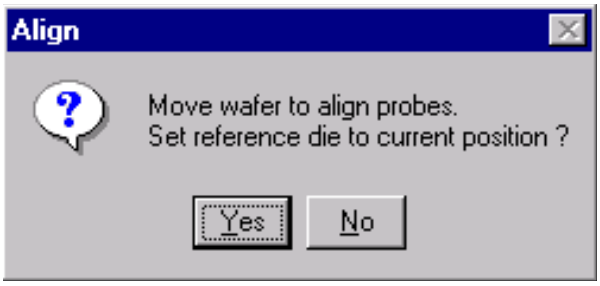
9. From the align dialog box, check **Move to calculated position** from the Set Reference Die group (Figure K-38).

Figure K-38
Move the Reference Die



10. Click **OK**.
11. Manually move the wafer to the Reference Die. Then click **Yes** to set the reference die to the present position (Figure K-39).

Figure K-39
Align dialog



NOTE: *When choosing the reference die:*

1) The wafer should be on the chuck and physically in the correct REFERENCE position.
2) Click the die on the wafer map GUI that will be the reference die.
3) An R appears when a die has been selected as the home die.

---

12. From the **Nucleus UI** toolbar, click the **Hard Align** button (Figure K-40) to display the Hard Align dialog.

Figure K-40
Hard Align button

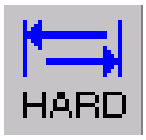
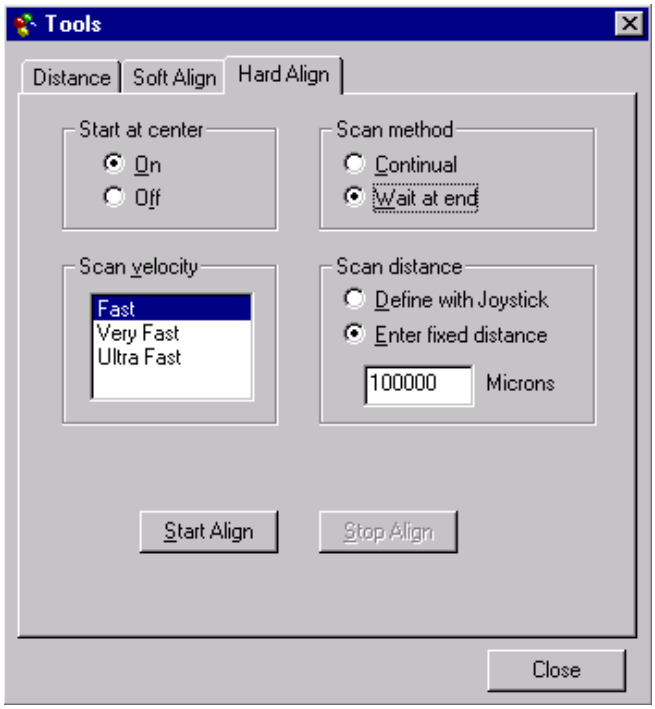
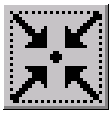


Figure K-41
Hard Align tab



13. On the Hard Align tab:
    - Check Start at the center **ON**.
    - Scan method as **Wait at end**.
    - Set desired scan velocity.
    - Enter the fixed scan distance or check the **Define with Joystick** box.
14. Align the wafer using the following steps:
    - Move to wafer center by clicking the **Center** button (Figure K-42) on the **Motion Control window**.
    - Click **Start Align** on the Hard Align dialog.

Figure K-42
Center button



NOTE: *Raise the platen arm if prompted (a prompt will only appear if the platen arm is down when you start the alignment).*

- Watch on the monitor while the stage moves down the street to position the needles near the left edge of the wafer.
- Adjust the theta knob on the stage while moving across the wafer.
- Click **Yes** at the prompt that appears on the screen.
- Watch on the monitor and continue to adjust theta while moving down the street to position the needles near the right edge of the wafer.
- Make a small adjustment in theta when motion stops.

- Click **No** when the alignment is correct.

15. Set the contact position (set the current Z as contact position):

NOTE: *The Z contact position is the specified point where probe needles make contact with the wafer when using the **Raise / Lower** button. The **Raise / Lower** button is located on the left-hand side of the Nucleus toolbar. Click the button to toggle to the make-contact or break-contact position.*

NOTE: *Good contact occurs when the probe tips make contact with the probe pad accounting for the tolerances of the probe needles and wafer plus any additional overdrive. Overdrive is the additional Z motion of the probe needles relative to the wafer after the initial contact. Overdrive ensures tolerable contact resistance by causing the probe tips to scrub through test pad surface oxide.*

- Either using the **Z Up / Z Down** buttons on the Motion Control window (Figure K-31), or the joystick if set for Scan Z Axis (see **CAUTION**), make contact with the wafer.
- When probe tips are making good contact with the wafer, right click on the **Contact** button.
- Click the **Set to Current Position** button (Figure K-43).

CAUTION **When the Joystick mode is set to "Scan Z Axis," the joystick will control Z movement (see the Mode drop-box in Figure K-31). While in this mode, the prober beeps providing an audible alert. When this alert is heard, care should be exercised when using the joystick for Z travel adjustments. Avoid damage to the probe needle or the wafer while changing the Z height.**

- The **Up / Down arrows** may be used to set Z contact. When using the arrows, travel is fast (coarse adjustment) when away from the Z contact position, and slow (fine adjustment) when close to the Z contact position.
- When setting the Z contact, the camera stays focused on the probe needles (not on the wafer).

Figure K-43
Set to Current Position button

Probesites KITE Project example

The following is a step-by-step procedure to properly configure the SUMMIT 12000 so the probesites KITE project executes successfully.

Nucleus UI

NOTE: *The following configuration is accomplished using Nucleus UI on the probe station PC.*

- Edit and open a wafer map file as described in Step 2. Set up wafer geometry and Step 3. Create a site definition and define a probe list earlier in this appendix.

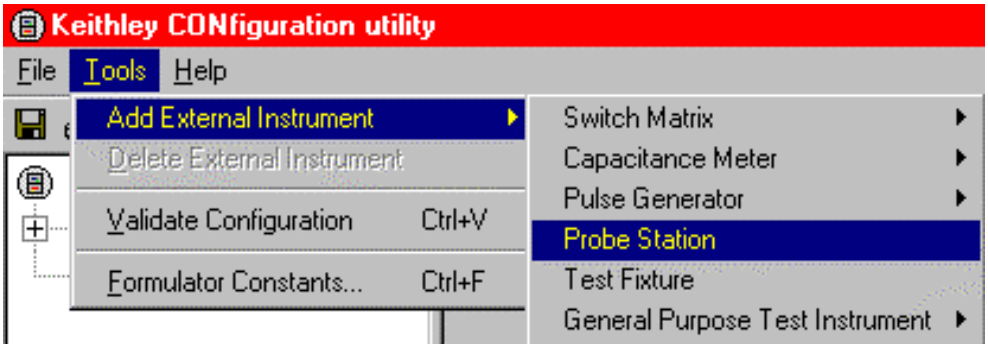
KCON

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KCON to add the prober to the configuration:

1. From the **Tools** menu (on Keithley CONfiguration Utility window), click **Add External Instrument Probe Station** (Figure K-44). The probe station **Properties** tab appears.
2. Select the cascade prober as the model (Figure K-45). Make sure the **Number of Pins / Positioners** is correct.

Figure K-44
KCON: Adding a prober



3. Save the configuration from the **File** menu (**Keithley CONfiguration Utility window**) (Figure K-46).
4. Exit KCON.

Figure K-45
KCON: selecting a prober

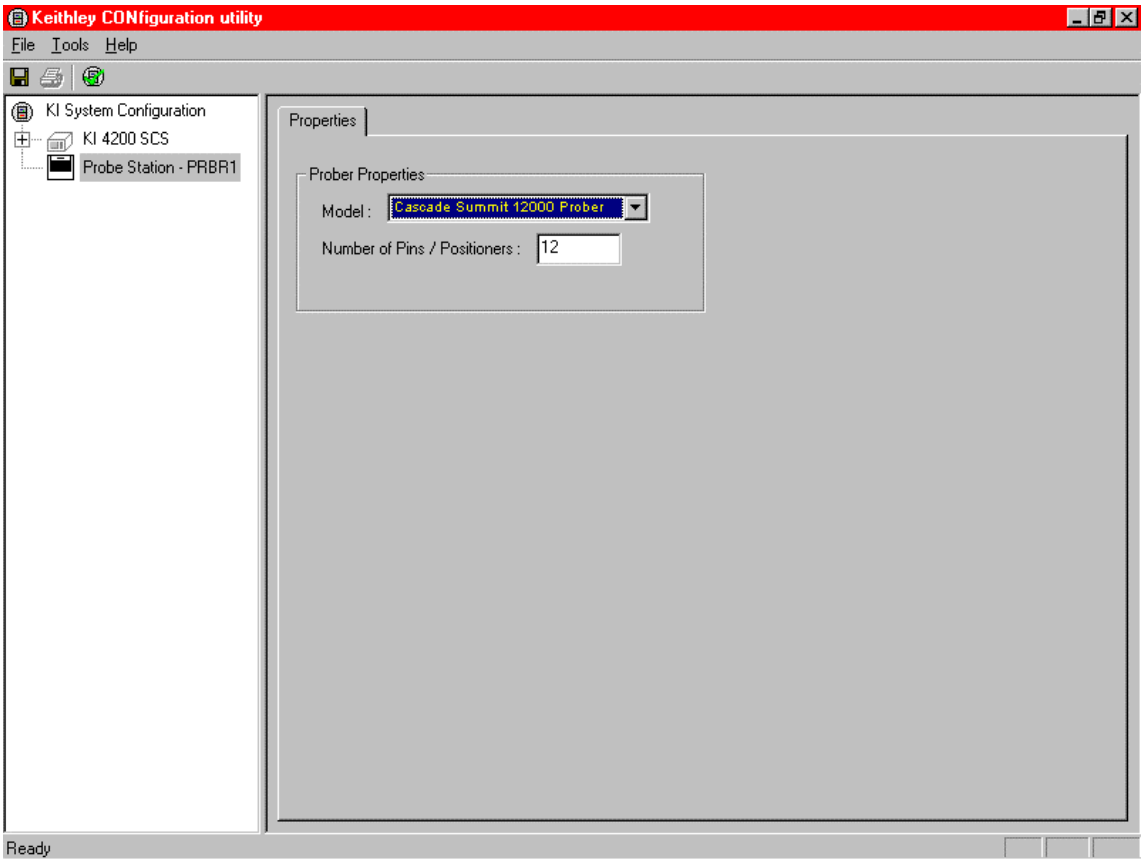
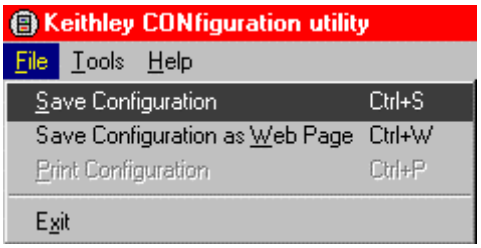


Figure K-46
KCON: Save Configuration



KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KITE to open and run the **probesites** project using the new configuration file, which will allow you to execute the project for this prober:

1. Open the **probesites** project example from KITE (Figure K-47):
   a. Run KITE.
   b. Select **Open Project** from the file menu.
   c. Open the folder: *c:\S4200\kiuser\Projects\probesites.*
   d. Select the project file: *probesites.kpr.*

Figure K-47
KITE: Open Project

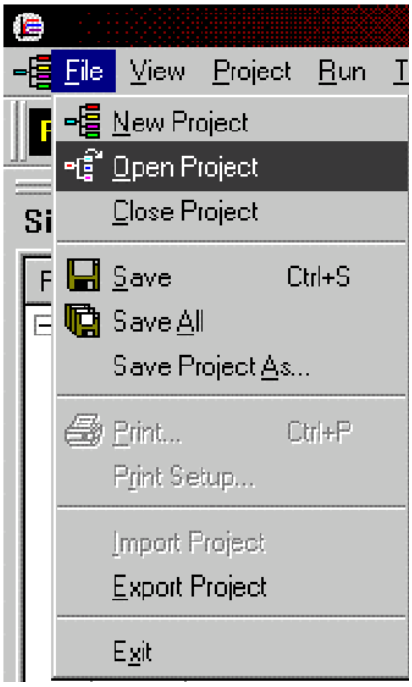
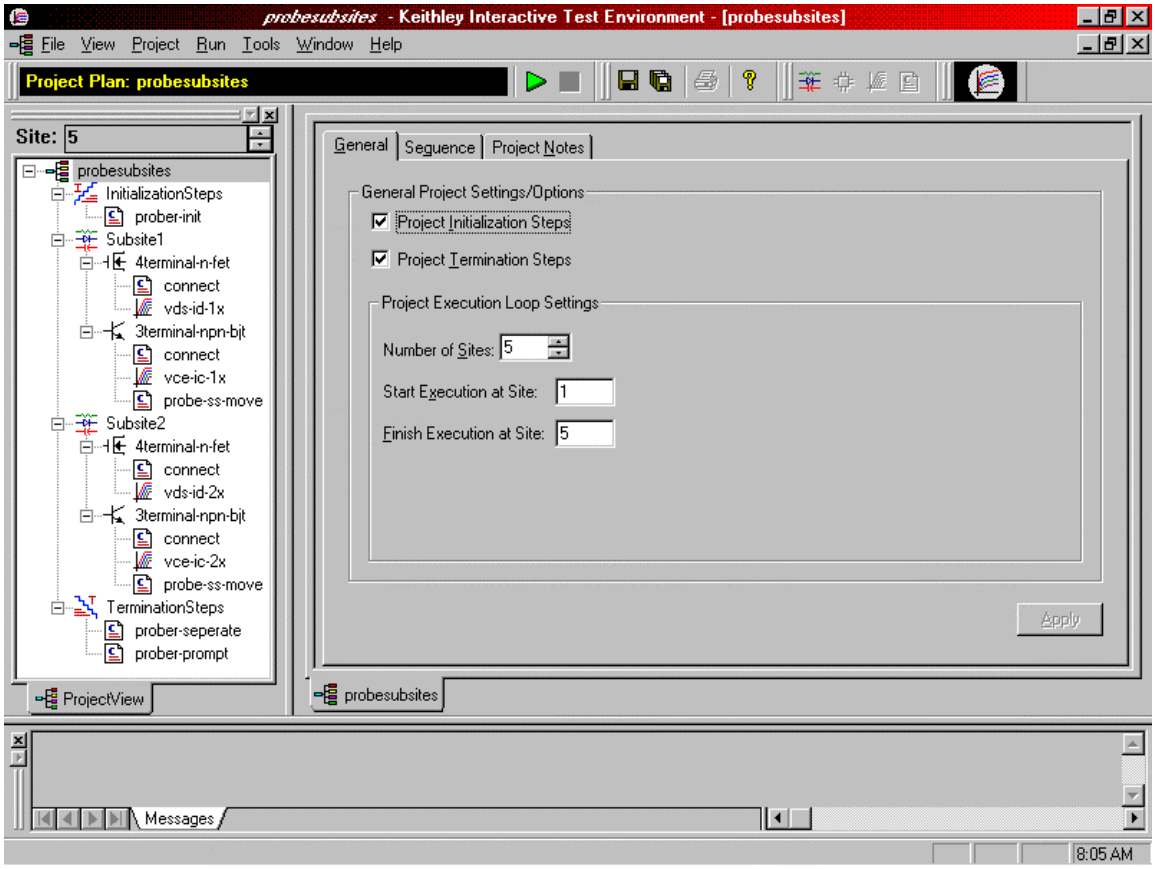


Figure K-48
KITE: probesites project



2. Select the top node in the **ProjectView** tab of the Project Navigator window.
3. Click the green **Run** button.

Probesubsites KITE Project example

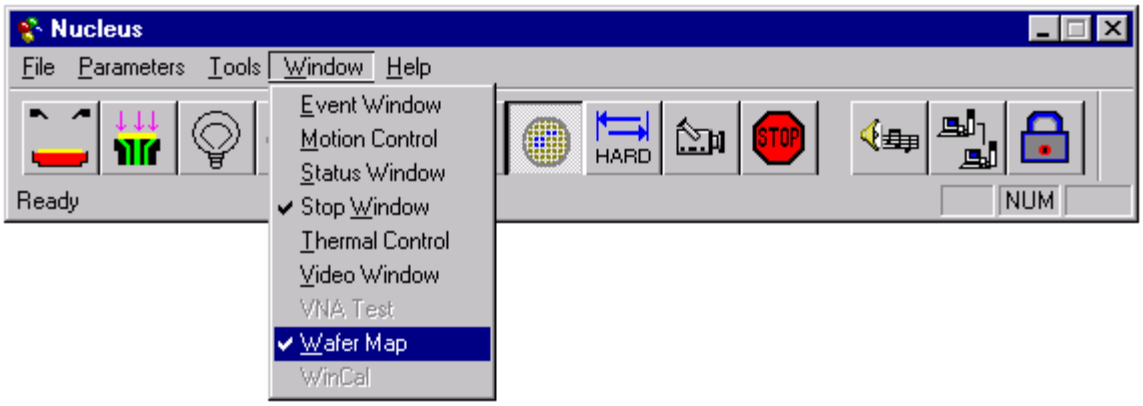
The following is a step-by-step procedure to properly configure the SUMMIT 12000 so the probesubsites KITE project executes successfully.

NOTE: *The following configuration is accomplished using the Nucleus UI on the probe station PC.*

1. Select the **Nucleus UI** icon on the desktop (Figure K-49).
2. Log in.
3. From the Nucleus UI toolbar, click **Windows** → **Wafer Map**.

Figure K-49
Nucleus UI icon

Figure K-50
Nucleus toolbar



4. From the **Wafer Map** window, select **File** → **Open** to open a wafer map file (Figure K-51).
5. Click **Wafer** → **Sub Die** from the Wafer Map menu (Figure K-52). A subsite dialog will appear.

Figure K-51
Wafer Map window

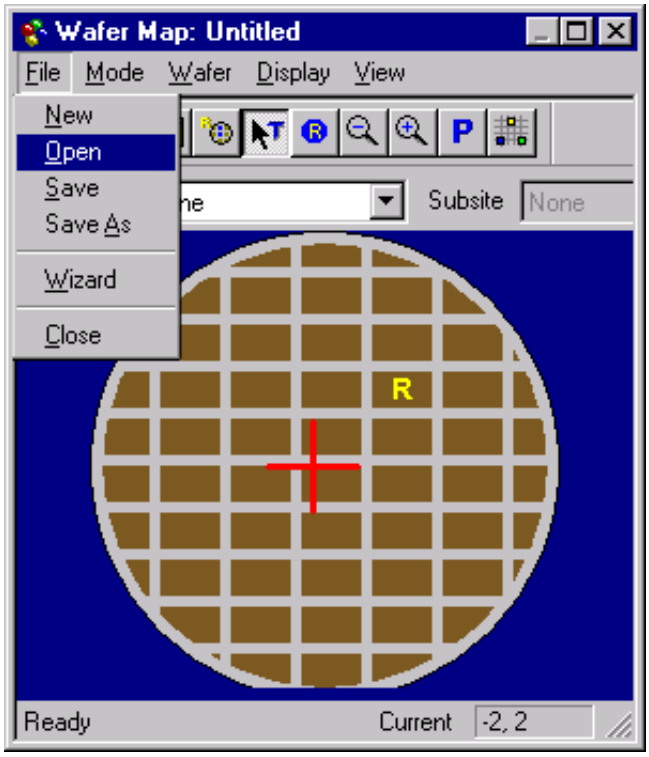
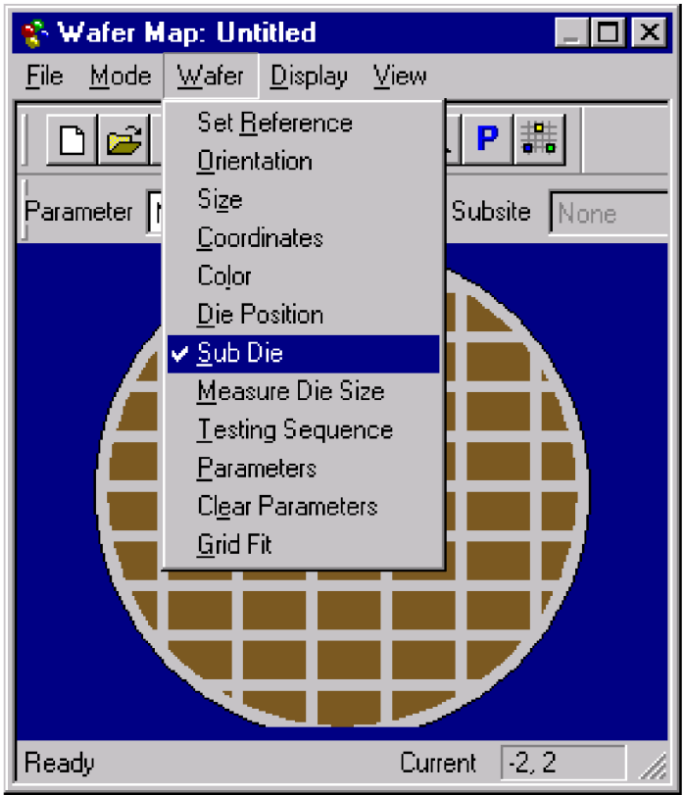
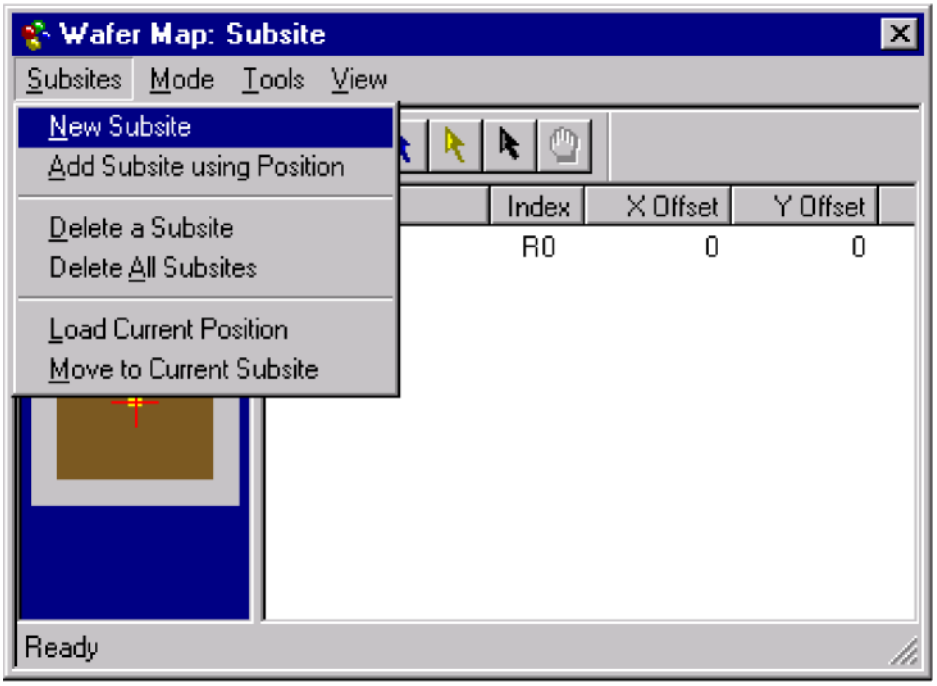


Figure K-52
Open Sub Die dialog



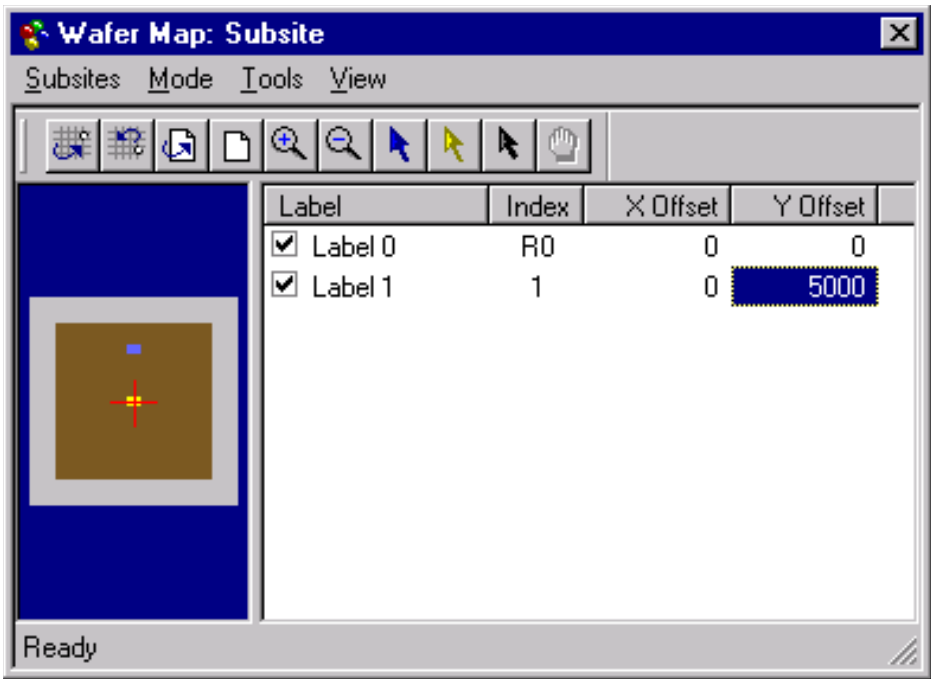
6. Click **Subsites** → **New Subsite** (Figure K-53). A new subsite Label 1 is created.

Figure K-53
Select New Subsite on the Subsites menu



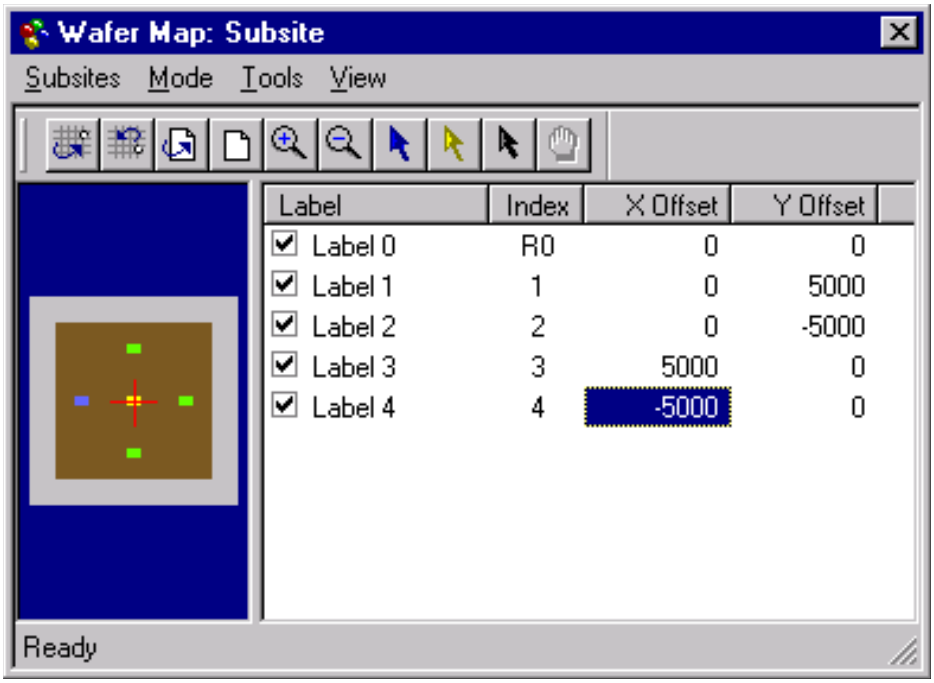
7. Enter the corresponding X and Y offset of the new subsite (Figure K-54).

Figure K-54
Enter x and y offset



8. Continue to add new subsites as desired until finished (Figure K-55).

Figure K-55
Make four new subsites



9. Click on the label name and type in a new description to relabel each subsite (Figure K-56).
10. To mark the subsite for testing, check the box at the front of each label. To skip testing the subsite, uncheck the box at the front of each label.
11. Click **File** → **Save** on the Wafer Map dialog (Figure K-51) to save the wafer map.

Figure K-56
Relabel the subsites

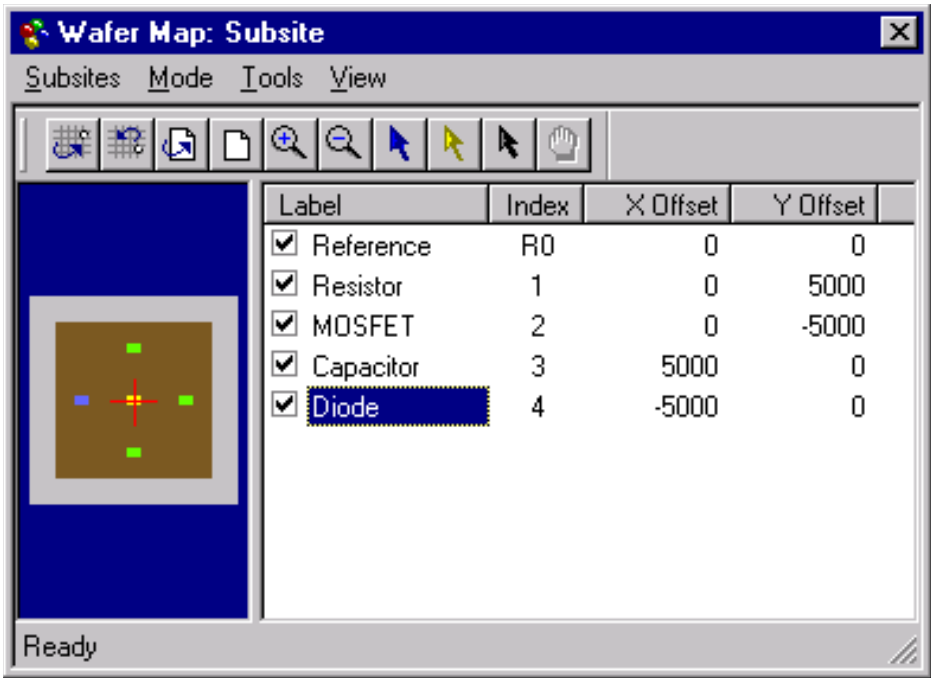
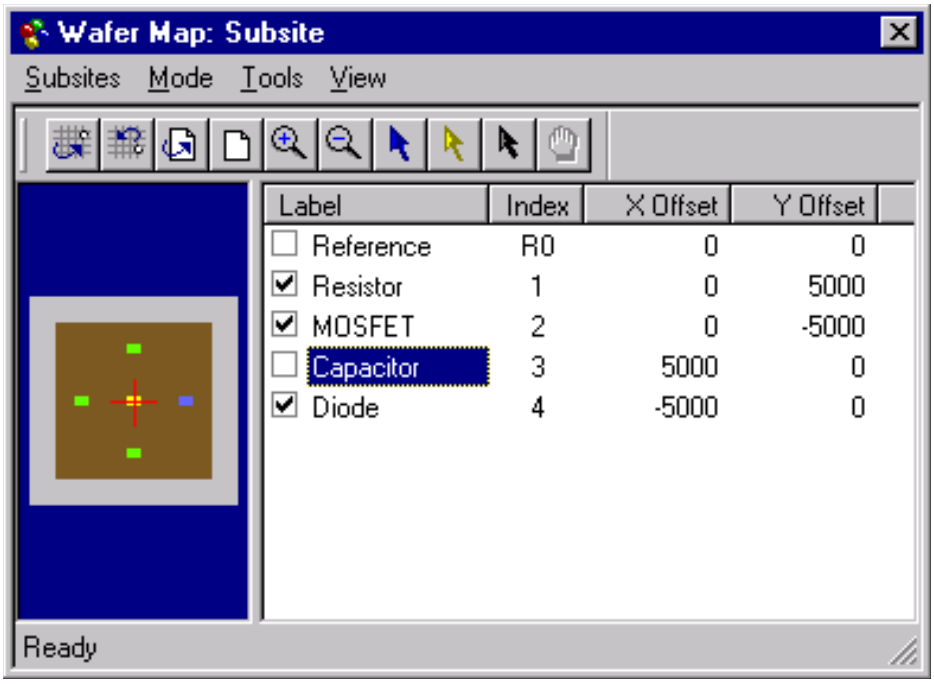


Figure K-57
Select sub die to test



KCON

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KCON to add the prober to the configuration.

Use the following procedure as a guide to add a prober to the Model 4200-SCS.

1. From the **Tools** menu (on Keithley CONfiguration Utility window), click **Add External Instrument > Probe Station** (Figure K-58). The probe station **Properties** tab appears.
2. Select the Cascade prober as the model (Figure K-59). Make sure the **Number of Pins / Positioners** is correct.

Figure K-58
KCON: Adding a prober

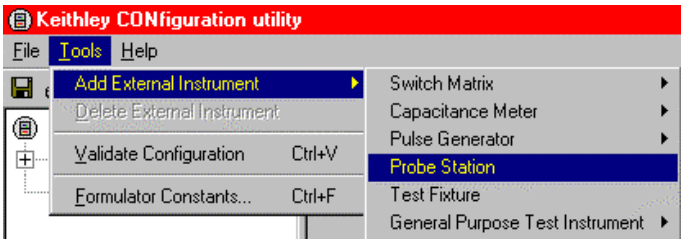
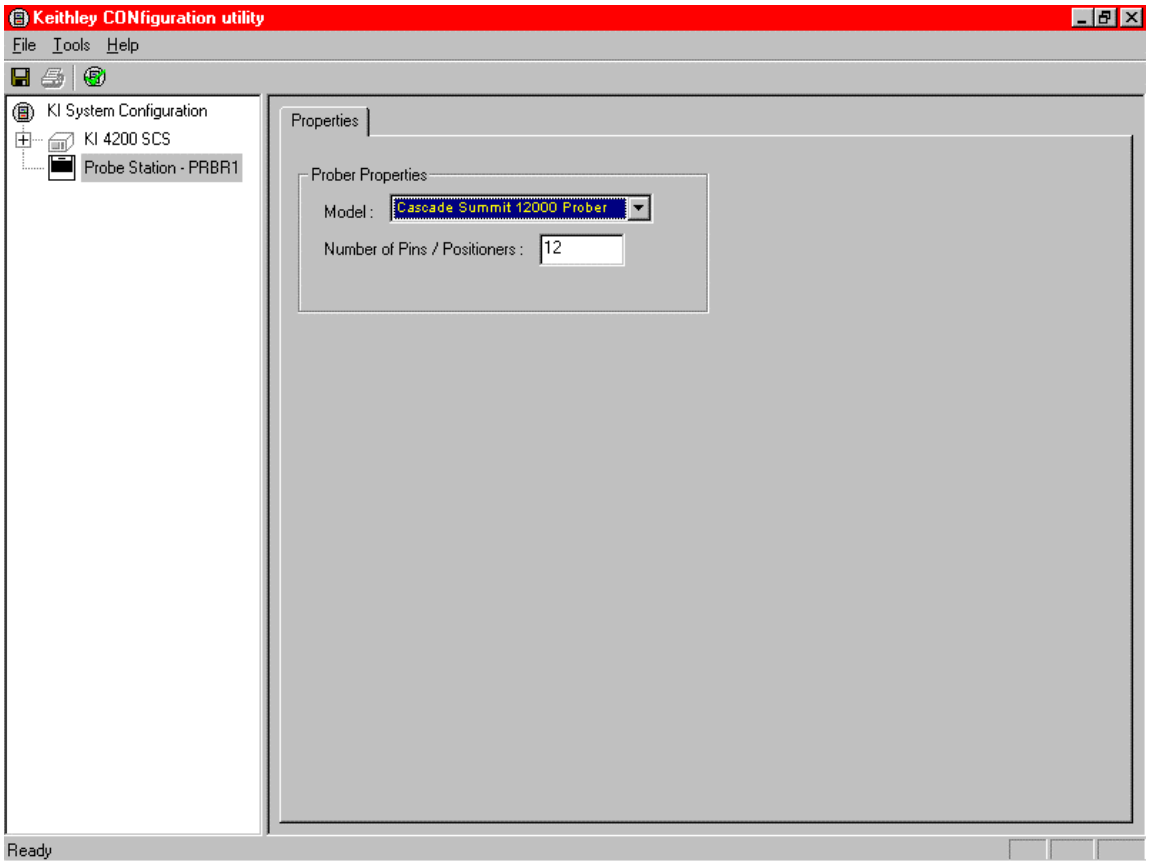
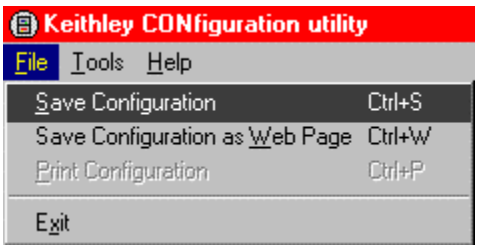


Figure K-59
KCON: Selecting a prober



3. Save the configuration from the **File** menu (Keithley CONfiguration utility window) (Figure K-60).
4. Exit KCON.

Figure K-60
KCON: Save Configuration



KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS computer.*

Use KITE to open and run the **probesubsites** project using the new configuration file, which will allow you to execute the project for this prober.

Open the **probesubsites** project example from KITE (Figure K-61):

a. Run KITE.
b. Select Open Project from the file menu.
c. Open the folder: *c:\S4200\kiuser\Projects\probesubsites.*
d. Select the project file: *probesubsites.kpr.*

Figure K-61
KITE Open Project

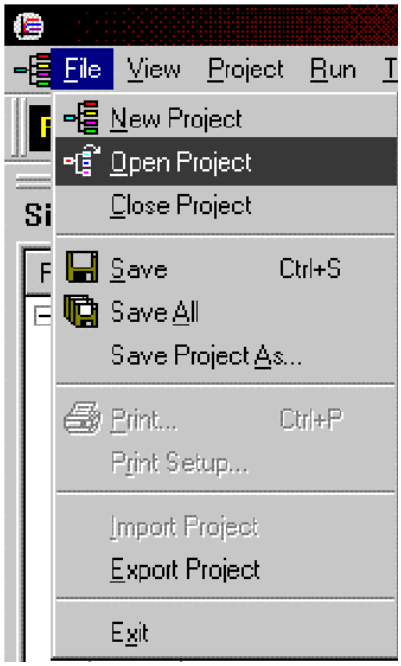
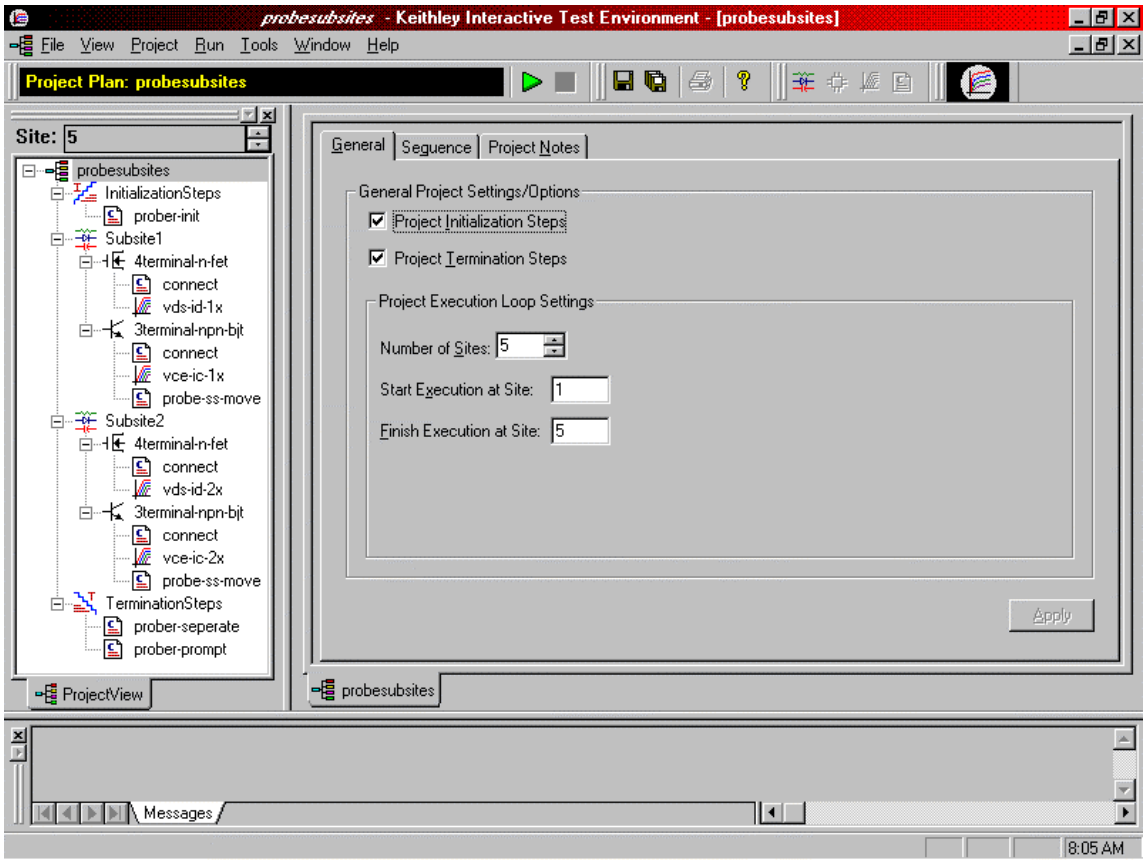


Figure K-62
KITE probesubsite project



1. Select the top node in the **ProjectView** tab of the Project Navigator window.
2. Click the green **Run** button.

Commands and error symbols

The following list (Table K-2) contains error and status symbols listed by command.

Table K-2
Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt
PR_OK	X	X	X	X
BAD_CHUCK	X			
INVAL_MODE	X			
UNINTEL_RESP	X	X	X	X
INVAL_PARAM		X		
BAD_MODE		X	X	X
UNEXPE_ERROR		X	X	X
PR_WAFERCOMPLETE			X	X

Appendix L
Signatone CM500 Prober

In this section:

Required probe station software

NOTE: *The CM500 driver with the Keithley Instruments Model 4200-SCS also works with other Signatone probers with Interlink controller such as the WL250 and S460SE. The name CM500 used in the configuration and setup in this section applies to all Signatone semi-auto prober system with Interlink controller.*

Software versions

The following software version was used to verify the configuration of the CM500 prober with the Model 4200-SCS: CM500.EXE ver. 2.5. For the S460-SE prober: S460SE.EXE ver. 2.5.

Probe station configuration

CAUTION **Although this appendix provides instructions on prober setup and configuration, refer to the Signatone CM500 or S460 Prober supporting documentation before attempting setup, configuration, or operation.**

There are four general steps required to setup and configure the CM500 or S460 prober for use with the Model 4200-SCS:

- Step 1. Set up communication
- Step 2. Set up wafer geometry
- Step 3. Load, align, and contact the wafer
- Step 4. Set up programmed sites and subsites

Each step is detailed later in this section.

Modifying the prober configuration file

NOTE: *This file is modified using the Model 4200-SCS.*

The default prober configuration file is contained in Figure L-1. As shown, the file is configured for use with a GPIB communication setup. Use Notepad and Microsoft® Windows® Explorer® to open, modify, and save this file should any change be required.

Configuration file location: C:\S4200\sys\dat\prbcnfg_CM500.dat.

Figure L-1
Sample CM500 prober configuration file

```
# prbcnfg_CM500.dat - DEFAULT Prober Configuration File
#
# The following tag, "PRBCNFG", is used by the engine in order to deter-
mine
# the MAX number of SLOTS and CASSETTES for a given prober at runtime.
#
<PRBCNFG>
#
# for OPTIONS "" == NULL, max 32 chars in string
#
# Example
#       01234567890
#PROBER_1_OPTIONS=1,1,1,1,1,1
#
#
#  OcrPresent
#  AutoAlnPresent
#  ProfilerPresent
#  HotchuckPresent
#  HandlerPresent
#  Probe2PadPresent
#
#
# Configuration for direct GPIB probers:
# CM500
#
PROBER_1_PROBTYPE=CM500
PROBER_1_OPTIONS=0,0,0,0,1,0
PROBER_1_IO_MODE=GPIB
PROBER_1_GPIB_UNIT=0
PROBER_1_GPIB_SLOT=1
PROBER_1_GPIB_ADDRESS=28
PROBER_1_GPIB_WRITEMODE=0
PROBER_1_GPIB_READMODE=2
PROBER_1_GPIB_TERMINATOR=10
PROBER_1_TIMEOUT=300
PROBER_1_SHORT_TIMEOUT=5
PROBER_1_MAX_SLOT=25
PROBER_1_MAX_CASSETTE=1
#
#
```

Step 1. Set up communication

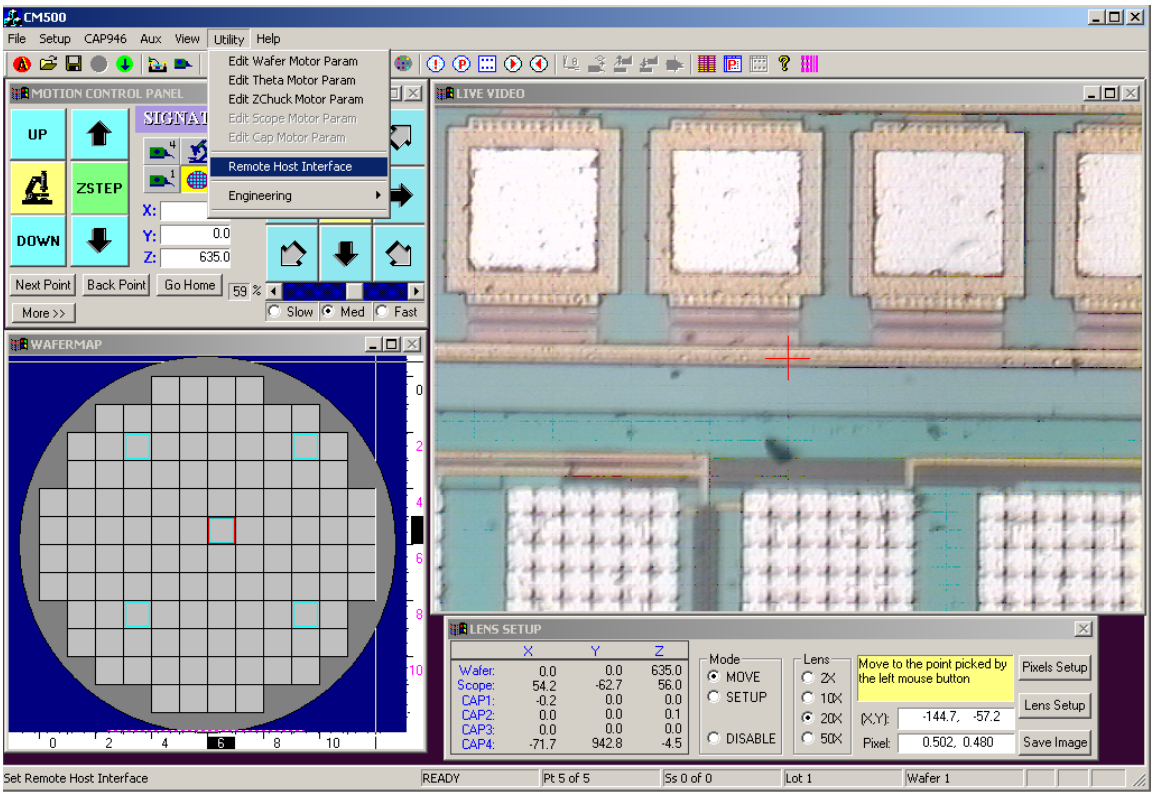
The Signatone CM500 prober is configured for GPIB communication only. Make sure the prober configuration is set up properly for the GPIB communication interface.

1. Double-click the **CM500** icon on the Windows desktop.

Figure L-2
CM500 icon

2. The prober will initialize the wafer XY stage, theta, and Z chuck.
3. Set up the GPIB interface.
   a. Select the **Utility** menu and run the **Remote Host Interface** command.

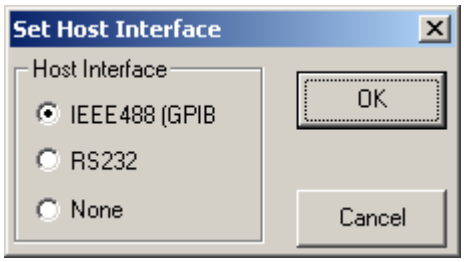
Figure L-3
CM500 Setup GPIB screen



b. Select the **IEEE488** Host Interface.

NOTE: *If IEEE488 has been enabled, press **Cancel** and skip the GPIB setup.*

Figure L-4
Select Host Interface



c. The Signatone GPIB driver window will show on the screen.

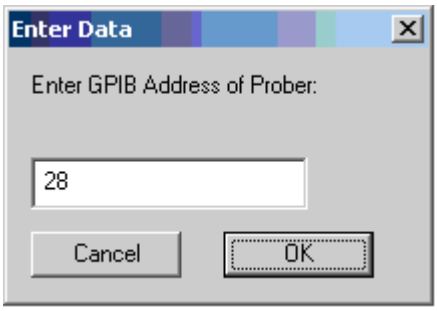
Figure L-5
Signatone GPIB driver window

d. Press **Addr** and check that the GPIB address matches the GPIB_Address setting in the S4200-SCS prober configuration file, `prbcnfg_CM500.dat` located at `C:\S4200\sys\dat`.

NOTE: *The default GPIB address is set to 28.*

e. If the address does not match, enter the new GPIB address, then press **OK**.

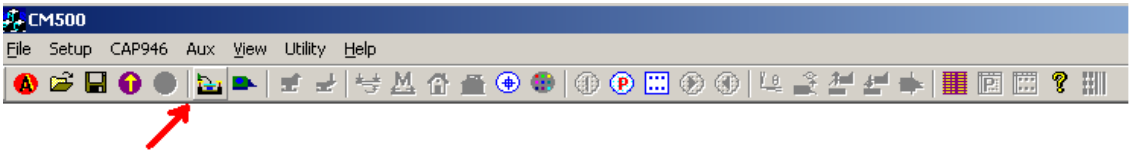
Figure L-6
Set GPIB Address



Step 2. Set up wafer geometry

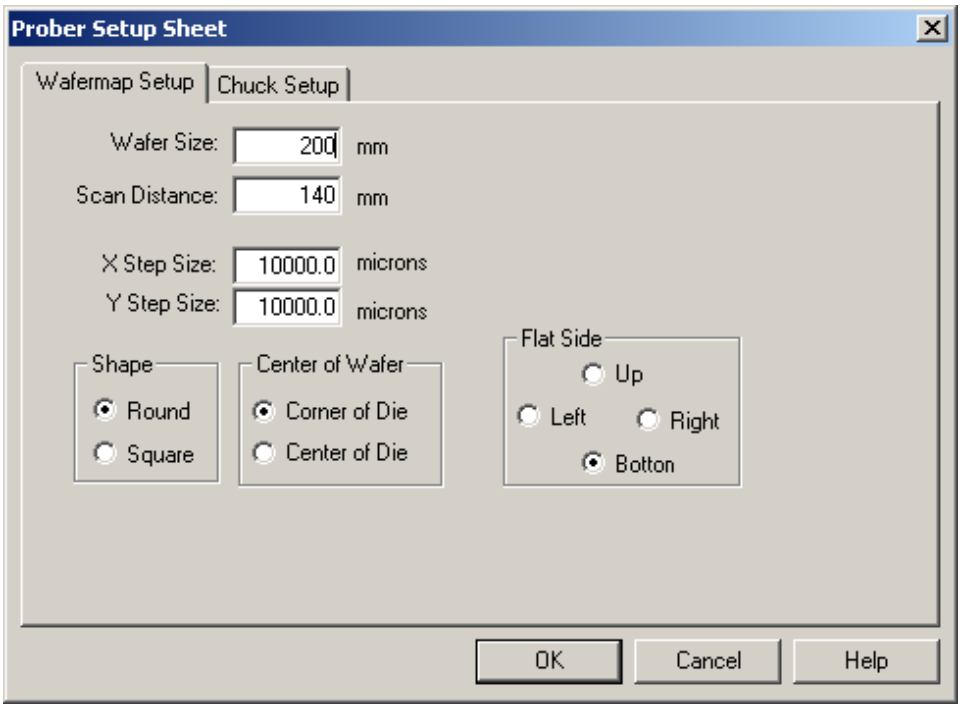
1. Press the **Prober Setup** icon on toolbar as shown below.

Figure L-7
CM500 Prober Setup icon



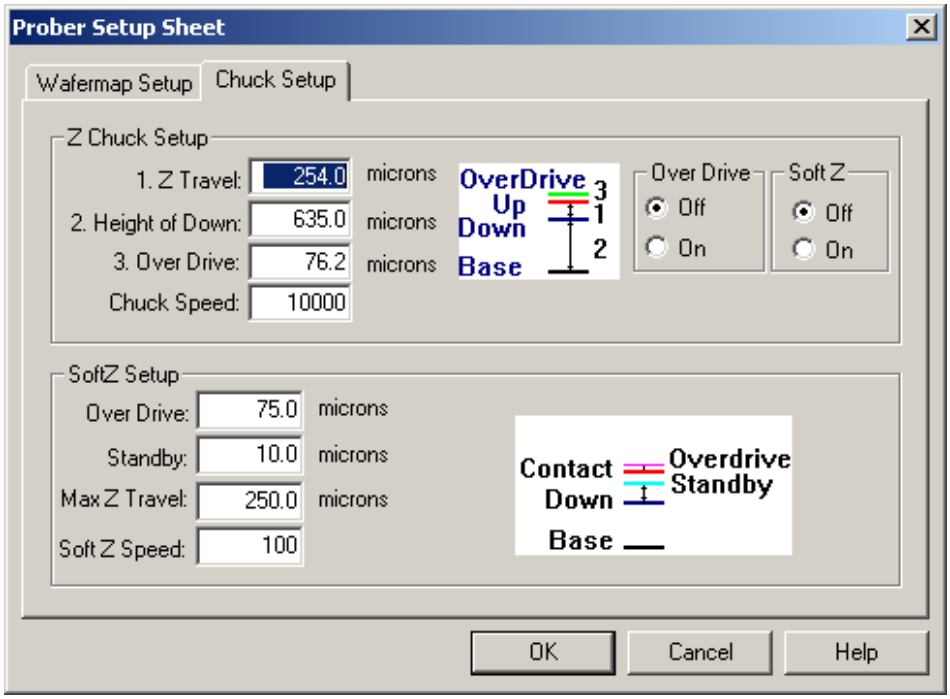
2. Select **Wafermap Setup** tab to setup wafer information such as wafer size, scan distance, X step size, and Y step size.

Figure L-8
CM500 Prober Setup Sheet



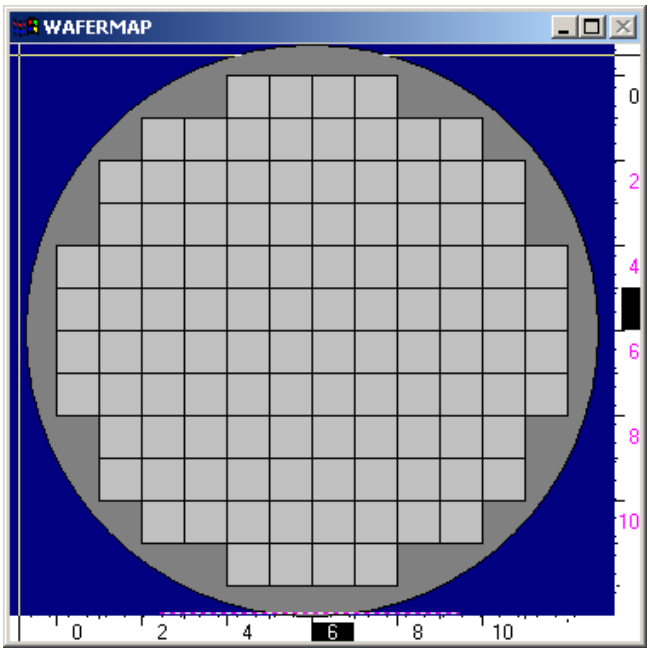
3. Select the **Chuck Setup** tab to enter Z chuck information such as Z travel and overdrive distance.

Figure L-9
CM500 Chuck Setup Sheet



4. After selecting **OK**, a new wafermap will show on the screen.

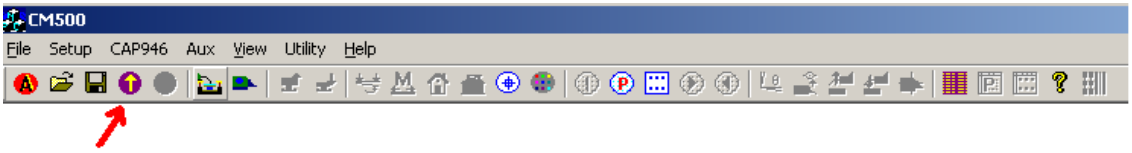
Figure L-10
CM500 Prober wafermap



Step 3. Load, align, and contact the wafer

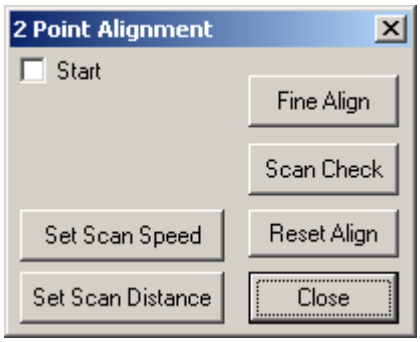
1. Press the **Load wafer** icon on toolbar.

Figure L-11
CM500 Prober load wafer icon



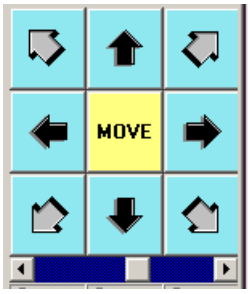
2. Select **Start** to move the wafer to Home and begin the sequences of 2 point alignment.

Figure L-12
CM500 Prober 2 Point Alignment 1



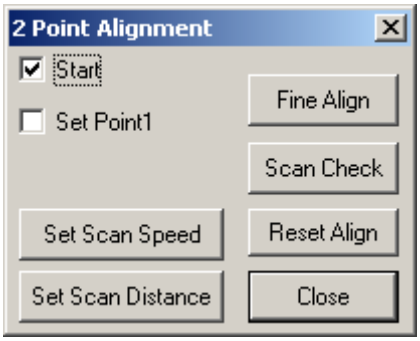
3. Press the **Arrow** buttons on the window (Figure L-13) to move the wafer stage to reference point 1.

Figure L-13
CM500 Prober manual MOVE buttons



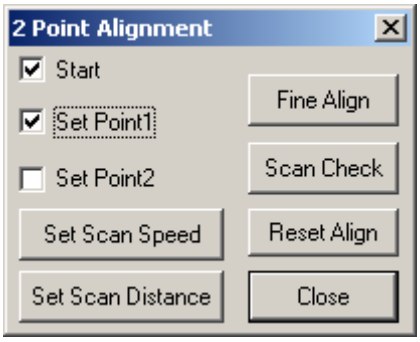
4. Check the **Set Point1** checkbox (Figure L-14).

Figure L-14
CM500 Prober 2 Point Alignment 2



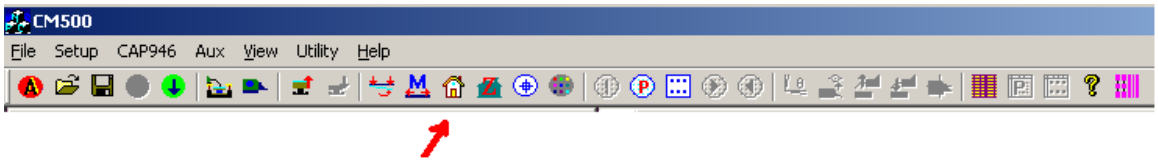
5. The Wafer stage will move to the other side as set by the scan distance.
6. Press the **Arrow** buttons on the window (Figure L-13) to move wafer stage to reference point 2.
7. Check the **Set Point2** checkbox (Figure L-15).

Figure L-15
CM500 Prober 2 Point Alignment 3



8. The Prober software will rotate the theta motor for the proper alignment.
9. Press the **Scan Check** button to check if the wafer is aligned well.
10. Press the **Fine Align** button for a further fine alignment.
11. After the wafer is aligned, set the HOME die of the wafer and wafermap.
    For the wafer:
    a. Move the wafer stage to the actual location that needs to be set as HOME.
    b. When completed moving wafer stage, press the **Set Home** icon on toolbar.

Figure L-16
CM500 Prober Set Home icon



For the wafermap:

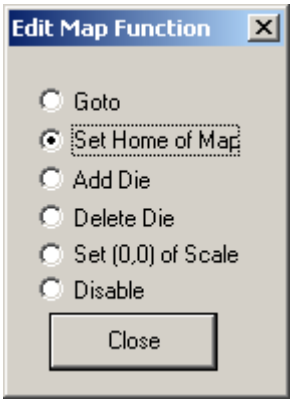
c. To set **HOME** on wafermap, press the **Edit wafermap** icon on toolbar.

Figure L-17
CM500 Prober Editmap function icon

d. Select the **Set Home of Map** function (Figure L-18).

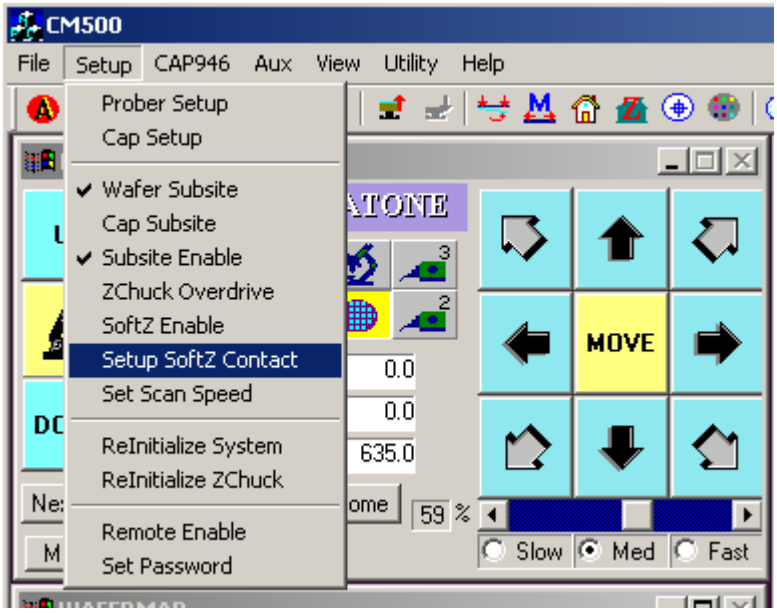
e. Click the Home die on wafermap that needs to be set as Home. Then **Close** the Edit map function window.

Figure L-18
CM500 Prober Edit Map Function window



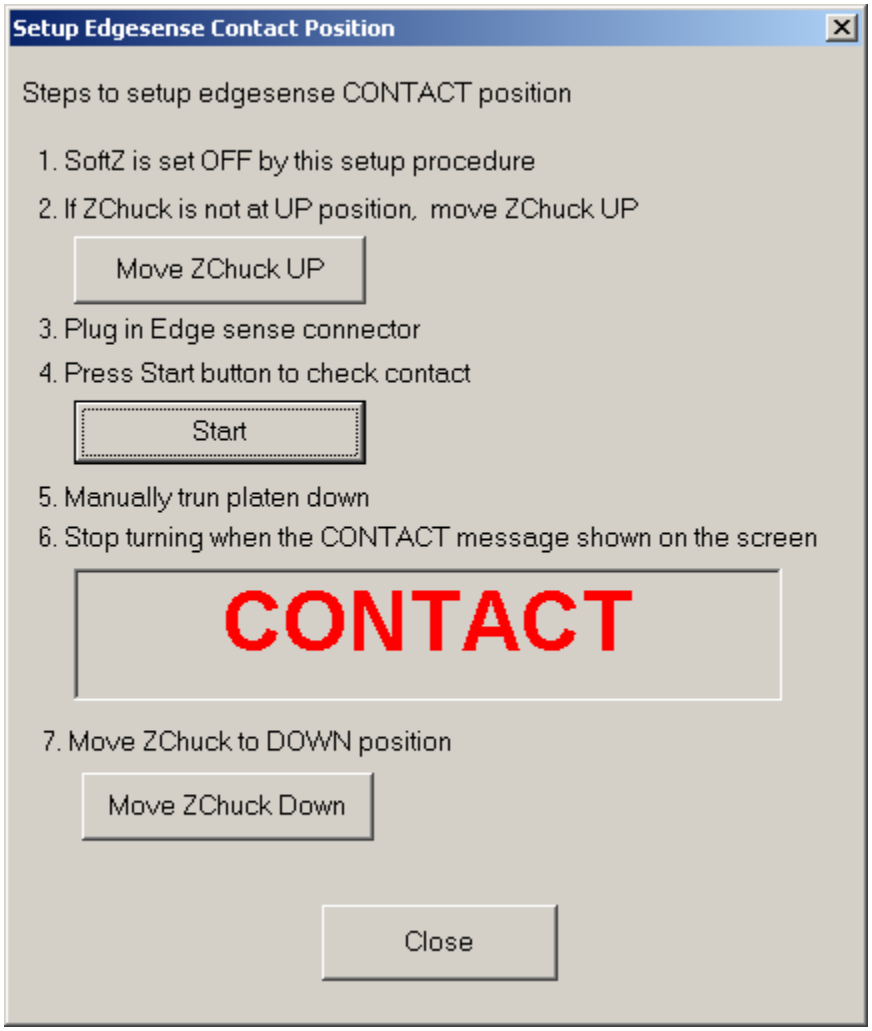
12. If an edge sense card is being used as the contact input for Z Chuck, the user must select the **Setup SoftZ Contact** option from the **Setup** menu.

Figure L-19
CM500 Prober Setup softz contact command



13. Follow the instructions on the window (Figure L-20) to adjust the height of platen and to determine the contact position of Z Chuck.

Figure L-20
CM500 Prober Setup Edgesense Contact Position window

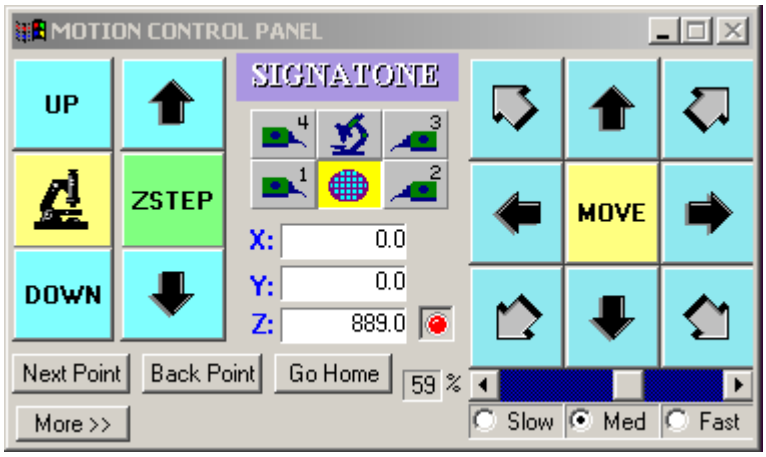


14. After completing the steps in Figure 20, move Z Chuck up to confirm contact condition using the **Contact** icon on toolbar (Figure L-21).

Figure L-21
CM500 Prober Z Chuck Up (CONTACT) icon

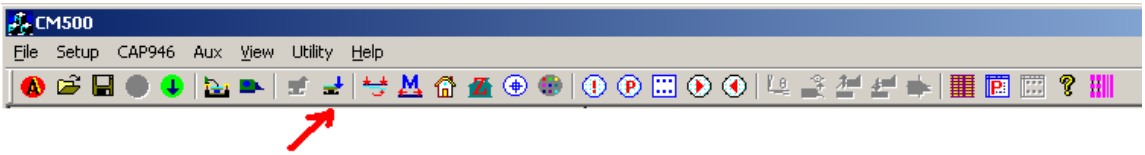
15. If the edge sense is plugged in for contact input, turn **ON** SoftZ. A red LED will appear in the motion control panel (See Figure L-22).

Figure L-22
CM500 Prober Motion Control Panel



16. Move Z Chuck down using the **Separate** icon on toolbar.

Figure L-23
CM500 Prober Z Chuck Down (SEPARATE) icon

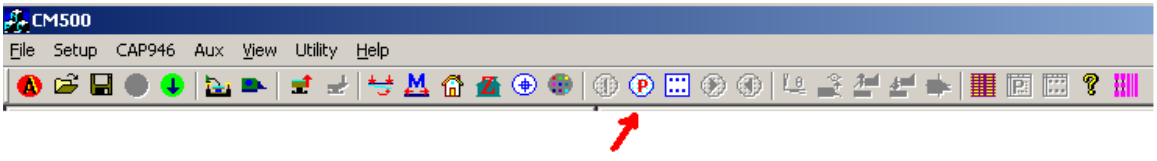


Step 4. Set up programmed sites and subsites

Case 1: Programmed sites without subsite setup

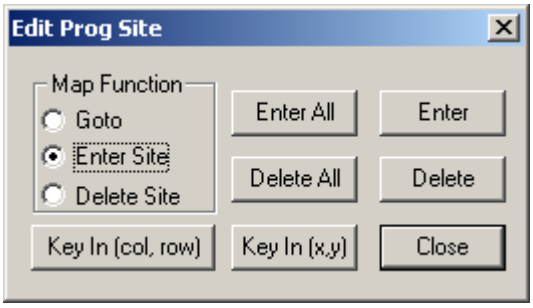
1. Pick the **Program Site** icon on toolbar (Figure L-24).

Figure L-24
CM500 Prober Edit Program Sites icon



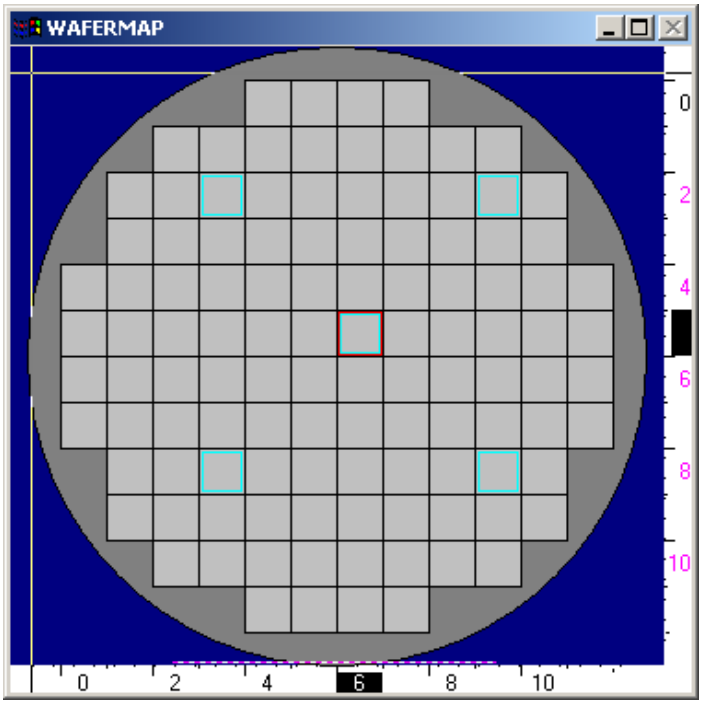
2. Select the **Enter Site Map** function.

Figure L-25
CM500 Prober Edit Program Site window



3. Move the mouse onto the WAFERMAP window and then either:
   a. Select the die(s) to be tested on wafermap and Press the **Enter** button
   OR
   b. Press the **Enter All** button to test all dies

Figure L-26
CM500 Prober wafermap includes program sites

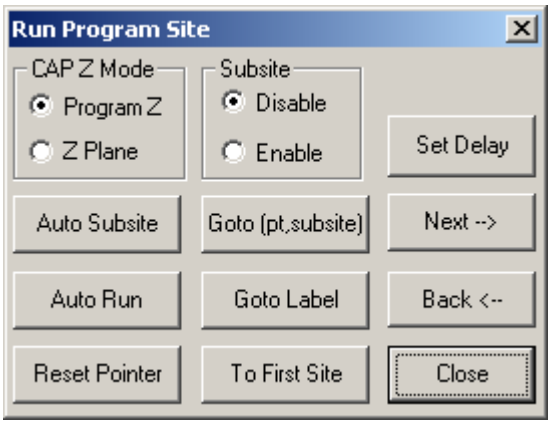


4. To step through all the programmed sites, select the **Run Program Site** icon on toolbar.

Figure L-27
CM500 Prober Run Program Sites icon

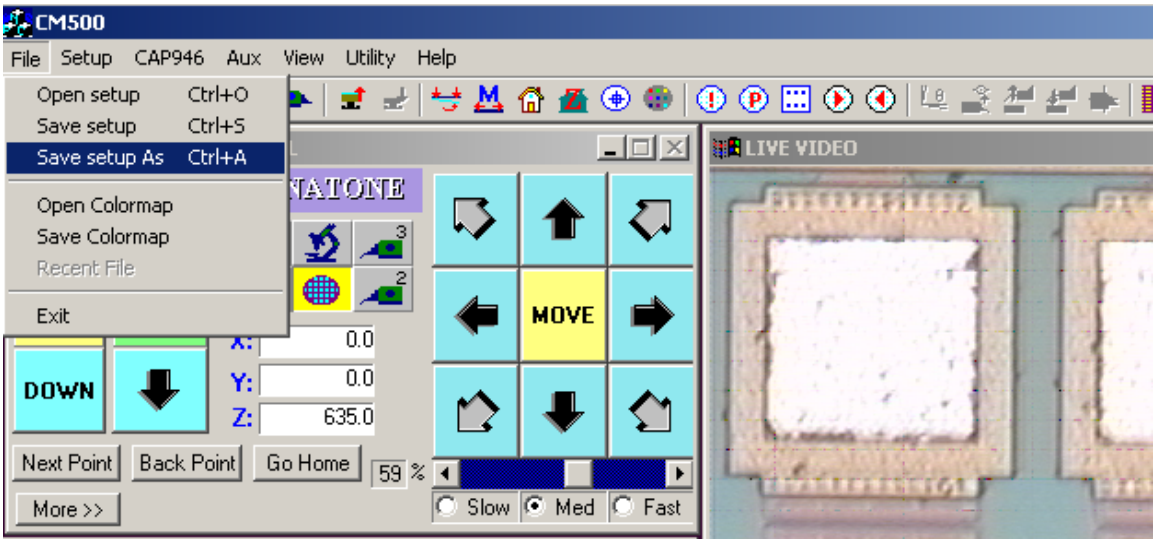
5. Press the **To First Site** button to move the prober to the first programmed site for testing. Make sure Subsite (template) is disabled here (Figure L-28).

Figure L-28
CM500 Prober Run Program Site window



6. In the **File** menu bar, select the **Save setup As** command to save the file to hard disk. The user can load this setup later without going through all the procedures again.
7. The Prober is now ready to accept remote command from S4200.

Figure L-29
CM500 Prober save setup



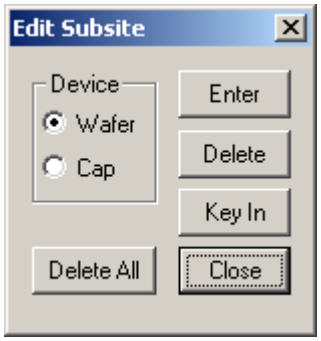
Case 2: Programmed sites with subsites setup

8. Pick the **Edit Subsite** icon on toolbar.

Figure L-30
CM500 Prober Edit Subsite icon

9. Select **Wafer** as the subsite device.

Figure L-31
CM500 Prober Edit Subsite window



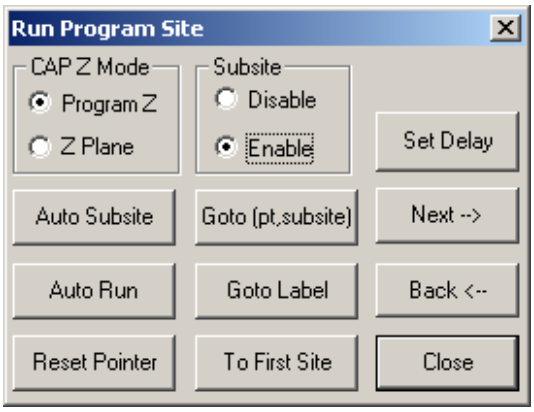
10. Move the wafer stage (see Figure L-22) to HOME position first.

NOTE: *All data recorded for the subsite are relative to the corner of the home die. The user can record the position of the subsite either by keying in the coordinates of the subsite using the keyboard, or by moving the wafer to the actual position and pressing **Enter**.*

---

11. To step through all the programmed sites and subsites, pick the **Run Program Site** icon (see Figure L-27) on the toolbar.
12. Make sure the Subsite (template) is **Enabled** (Figure L-32) if subsites are to be used. Then press the **To First Site** button to move wafer stage to first site of probing lists.

Figure L-32
CM500 Prober Enable Subsite



13. In the **File** menu bar, select the **Save setup As** command (see Figure L-29) to save the file to hard disk. The user can load this setup next time without going through all the procedures again.
14. The Prober is now ready to accept remote command from the S4200.

KITE project example for Probe Sites

The following is a step-by-step procedure to properly configure the KITE project to execute testing and auto wafer stepping to all programmed sites successfully. When the CM500 prober is connected to the Model 4200-SCS by GPIB interface, Model 4200-SCS is the GPIB master controller and CM500 is always in listening mode. The Model 4200-SCS will send control commands to CM500 to move prober to next site during the automatic testing. There are four interface commands: **PrInit, PrChuck, PrMovNxt,** and **PrSSMovNxt**. The user needs to add these commands into the KITE project.

CM500

NOTE: *The following configuration is accomplished using the CM500 on the probe station PC.*

Step 1. Follow the Probe Station Configuration Steps 2 to 4 (noted at beginning of this appendix).

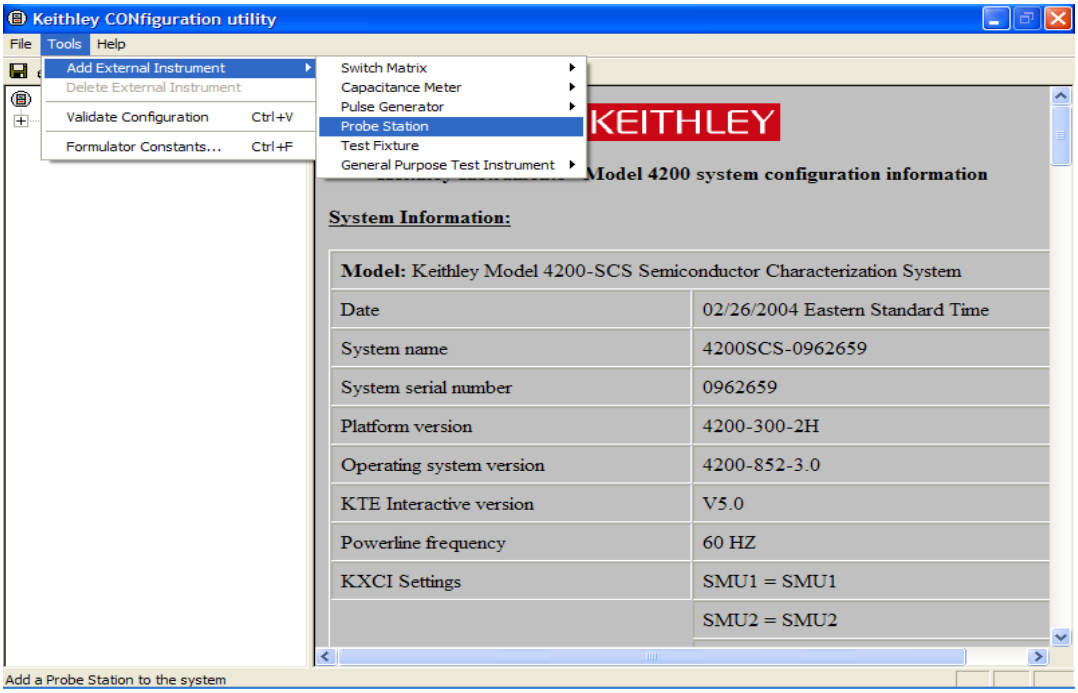
KCON

NOTE: *The following configuration is accomplished using the Model 4200-SCS. Use KCON to add the prober to the configuration.*

Add a prober to the Model 4200-SCS:

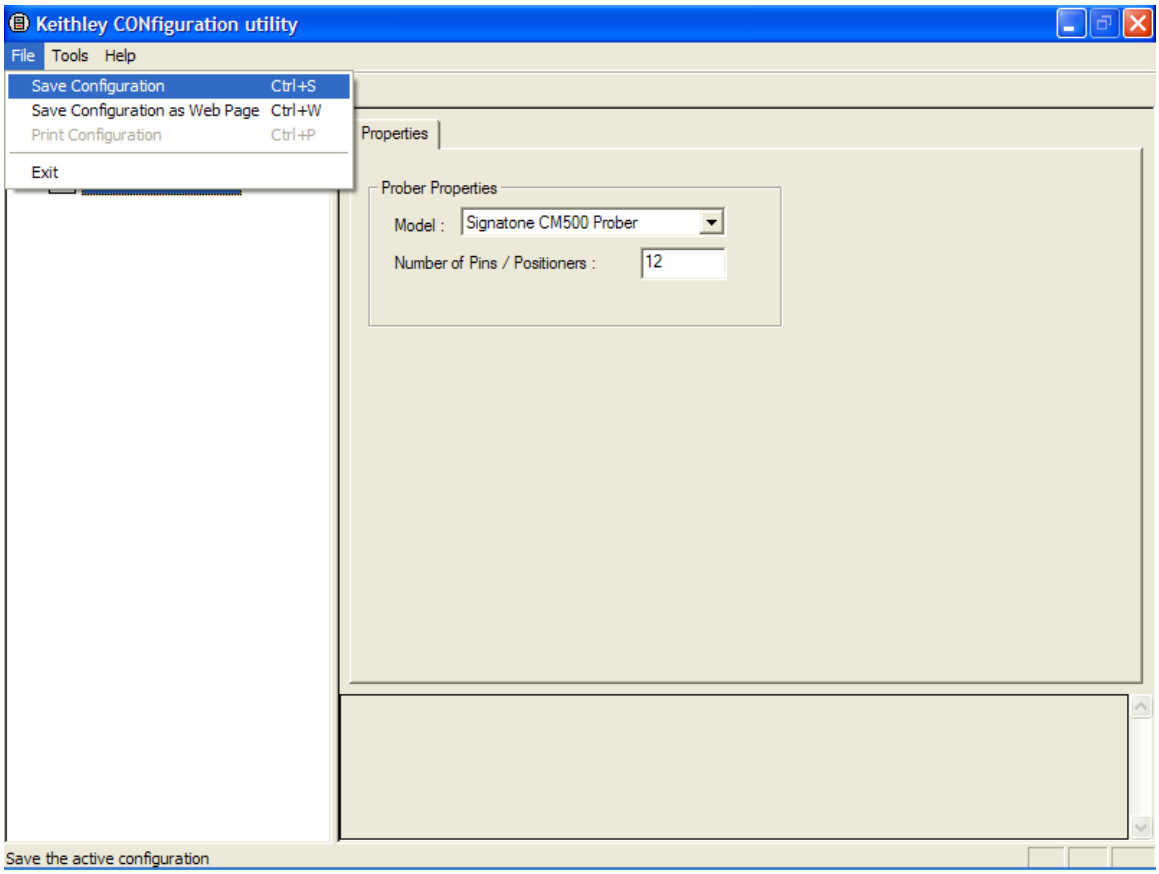
1. From the **Tools** menu (on the Keithley CONfiguration utility window), click **Add External Instrument > Probe Station**. Select the **Probe Station** property.

Figure L-33
Keithley KCON Tools



2. After the **Properties** tab appears, select the **Signatone CM500 Prober** as the **Model** and make sure the **Number of Pins / Positioners** is correct.
3. Save the configuration from the **File** menu (**Keithley CONfiguration Utility window**)
4. Exit KCON.

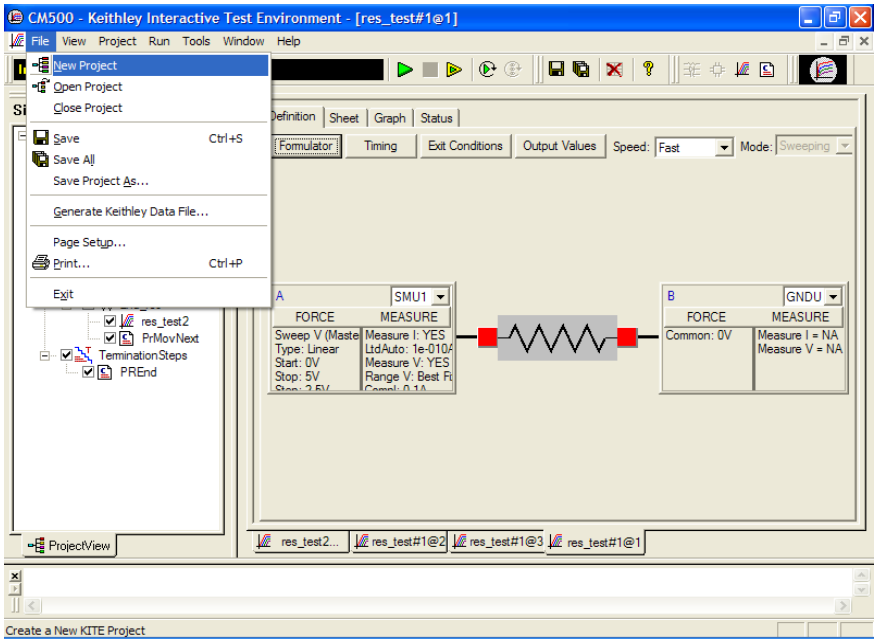
Figure L-34
Save Configuration



KITE project example

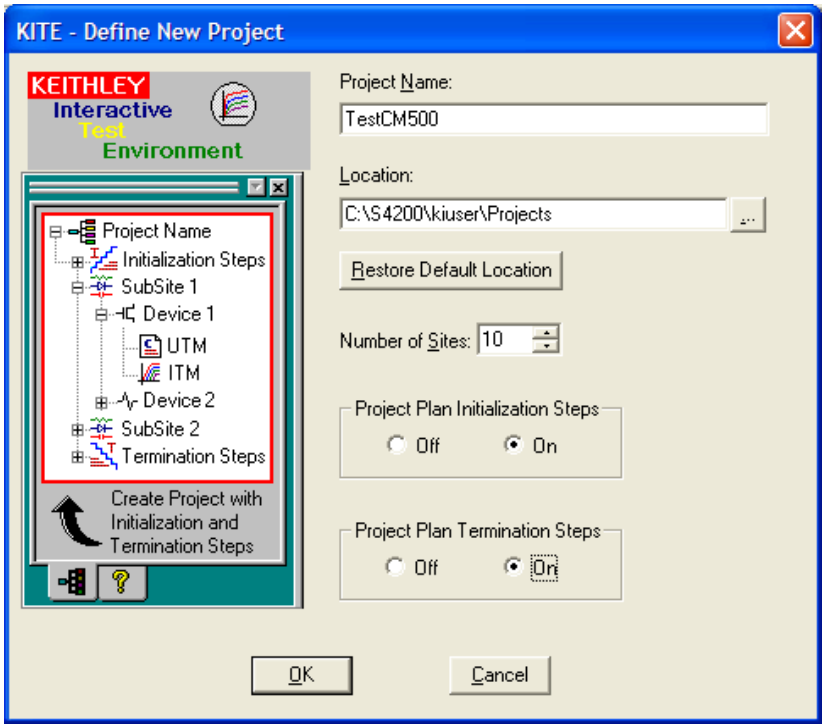
1. Run the KITE program from desktop. Open a new project by selecting **New Project** from the **File** menu.

Figure L-35
New Project



2. Enter the **Project Name** and **Location** fields.

Figure L-36
Set up Project Name



3. Under Initialization Steps: Select the **Add new UTM** icon, then select **prober-init**.

Figure L-37
Add a new UTM

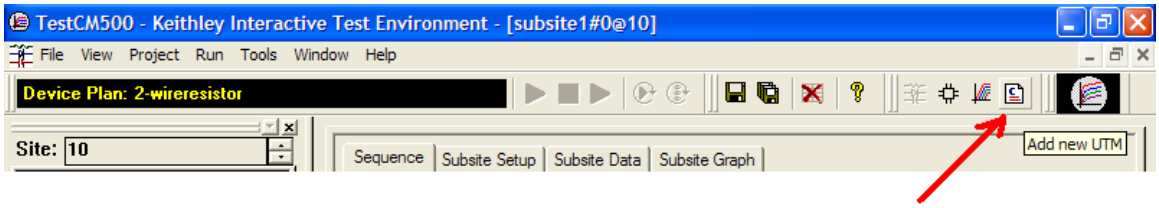
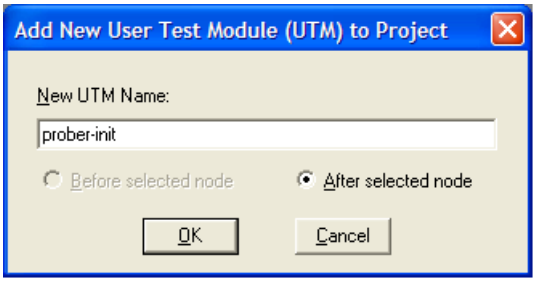
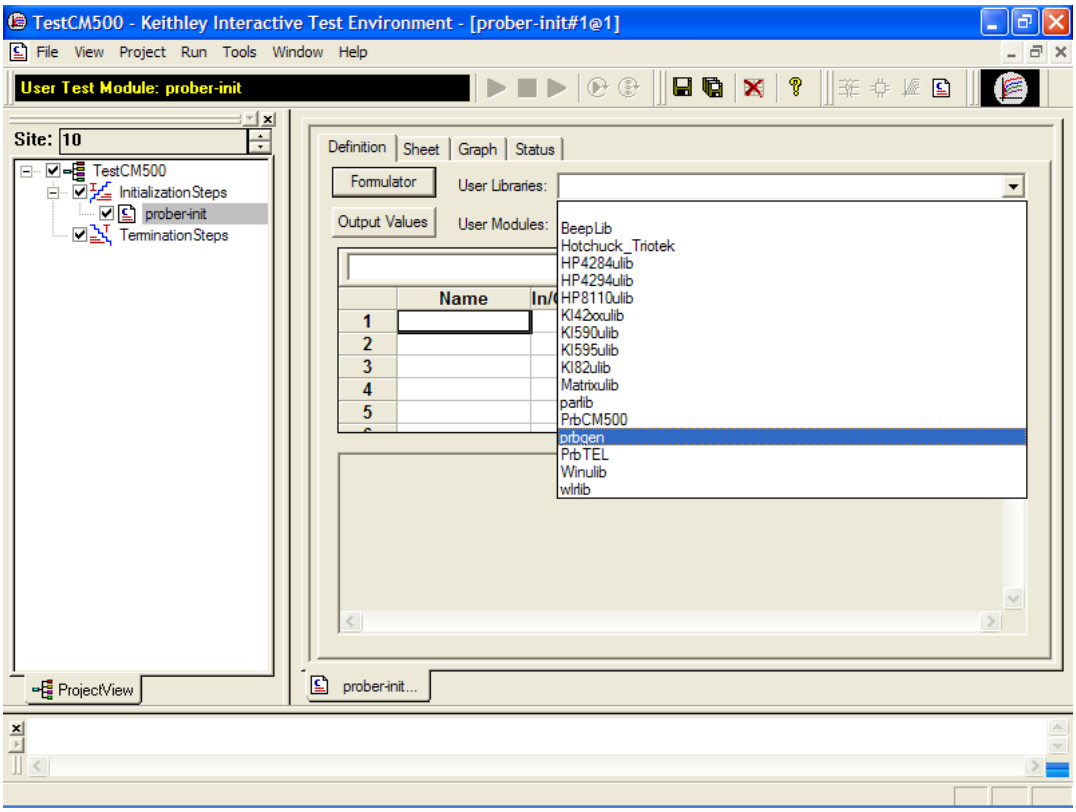


Figure L-38
Enter New UTM Name



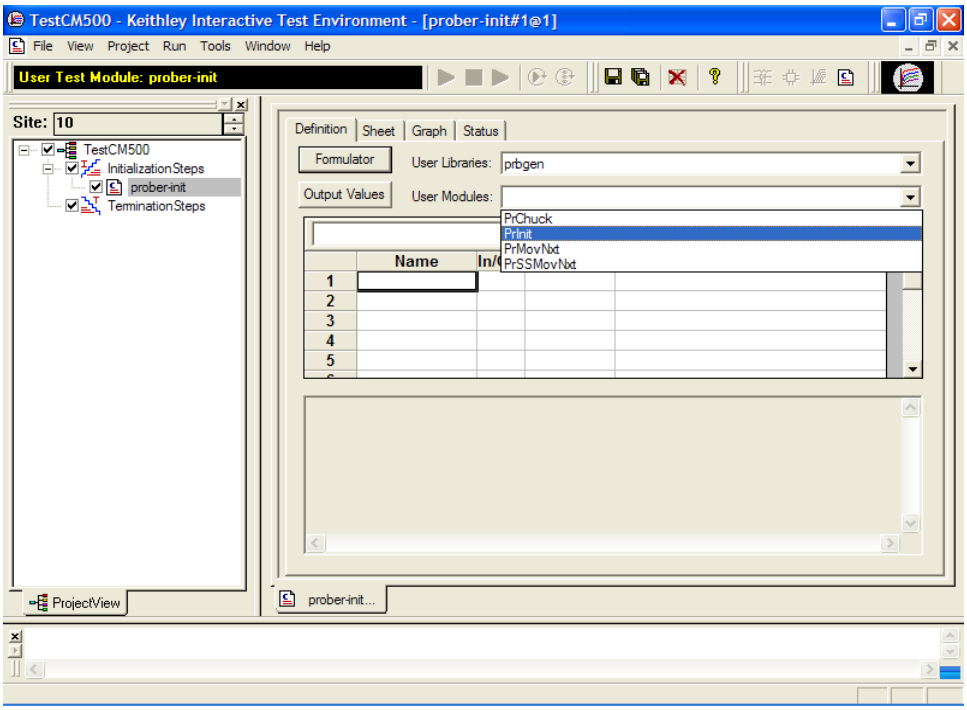
4. Select **prbgen** under the **User Libraries** drop down menu

Figure L-39
Select prbgen user library



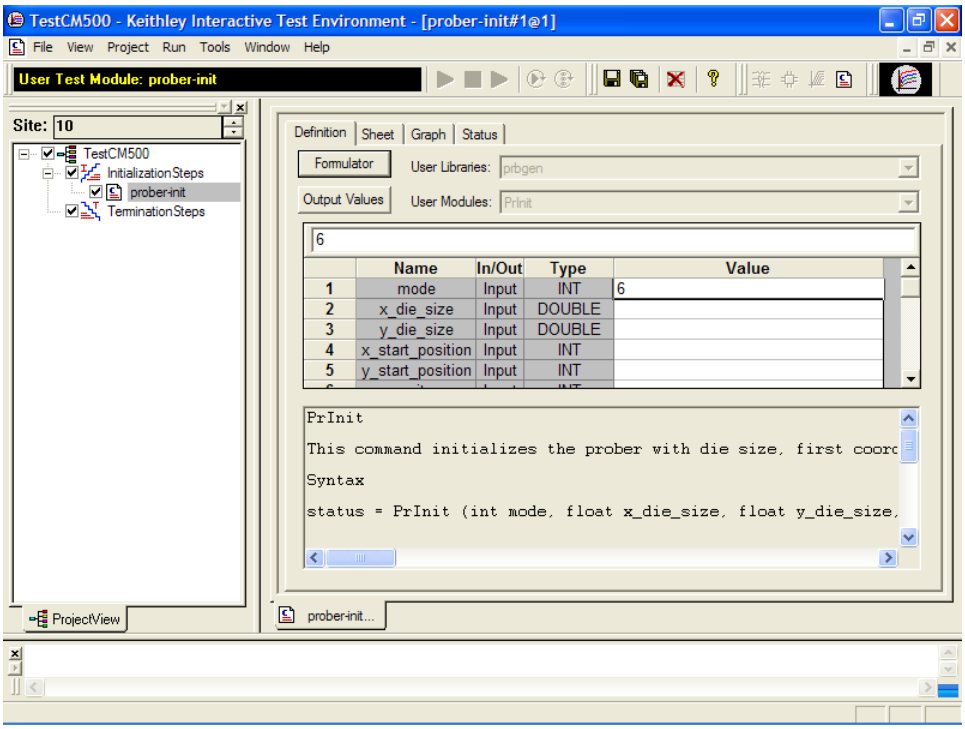
5. Select **PrInit** under the **User Modules** drop down menu.

Figure L-40
Select User Module



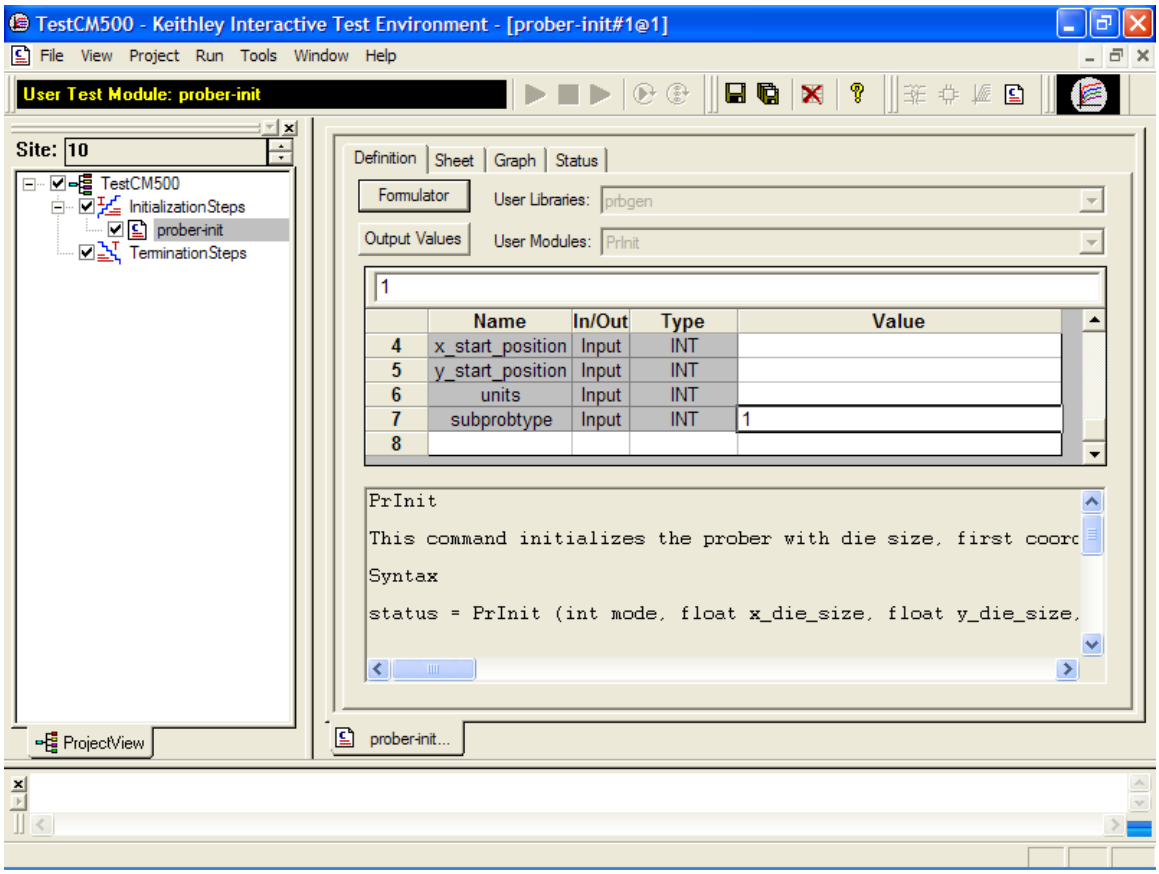
6. Set the first parameter of PrInit, which is **mode**, to 6 (see Figure L-41).

Figure L-41
Set PrInit parameters



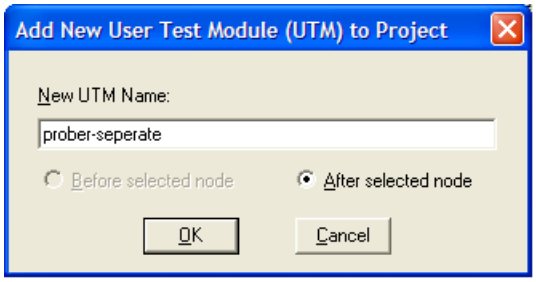
7. Check the last parameter's value. If the CM500 prober is currently not at its first site, set the last parameter subprobtype to 1; otherwise, set it to 0.

Figure L-42
Set subprobtype parameter



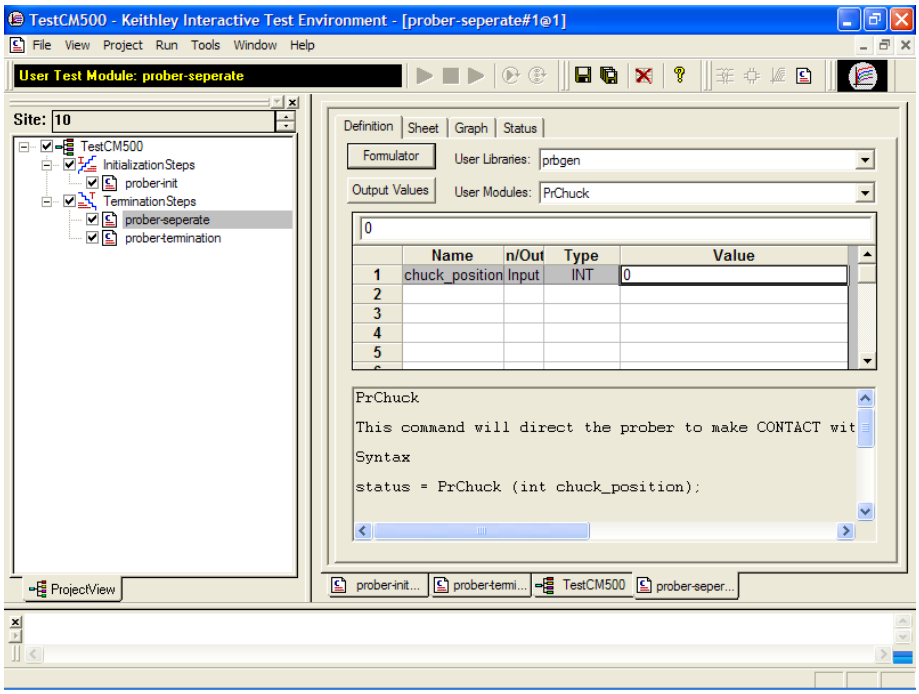
8. Under **Termination Steps**, add a new UTM named **prober-separate.**

Figure L-43
New prober-separate UTM



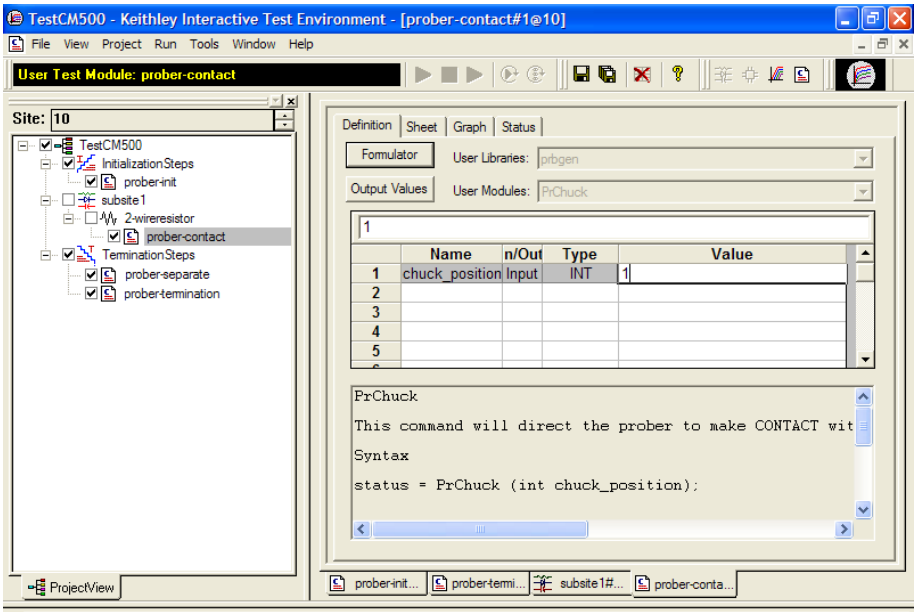
9. Select **PrChuck** under **User Modules** and set the **chuck_position** value to 0. This will move the Z Chuck to DOWN (separate) position.

Figure L-44
Set Chuck Down



10. Create a subsite and a device plan.
11. Add a new UTM named **Prober Contact** and set the **chuck_position** value to 1. This will move the ZChuck to CONTACT position.

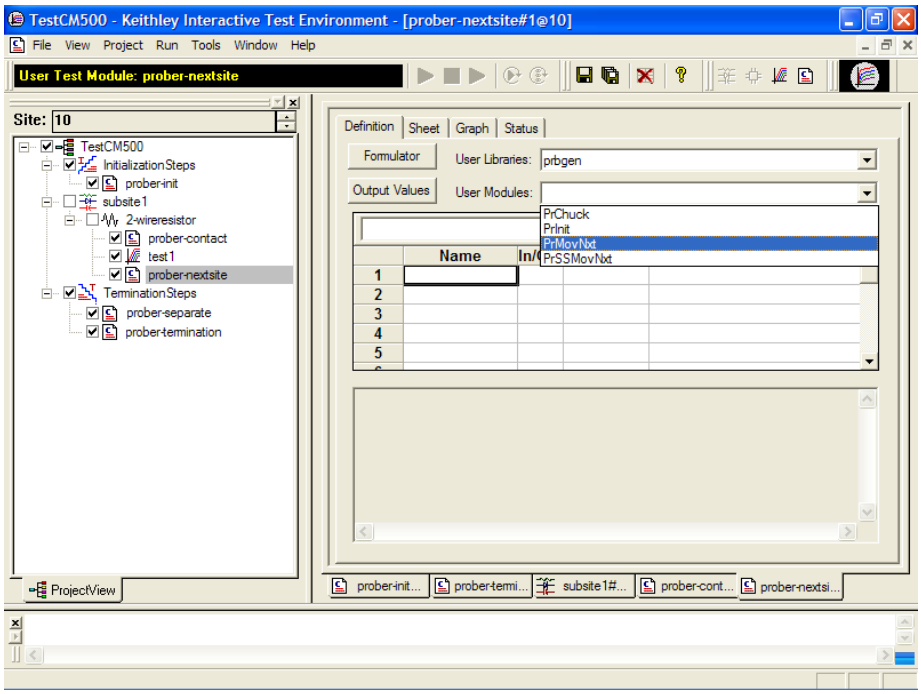
Figure L-45
Set Probe Contact



12. Add a new ITM for device testing.

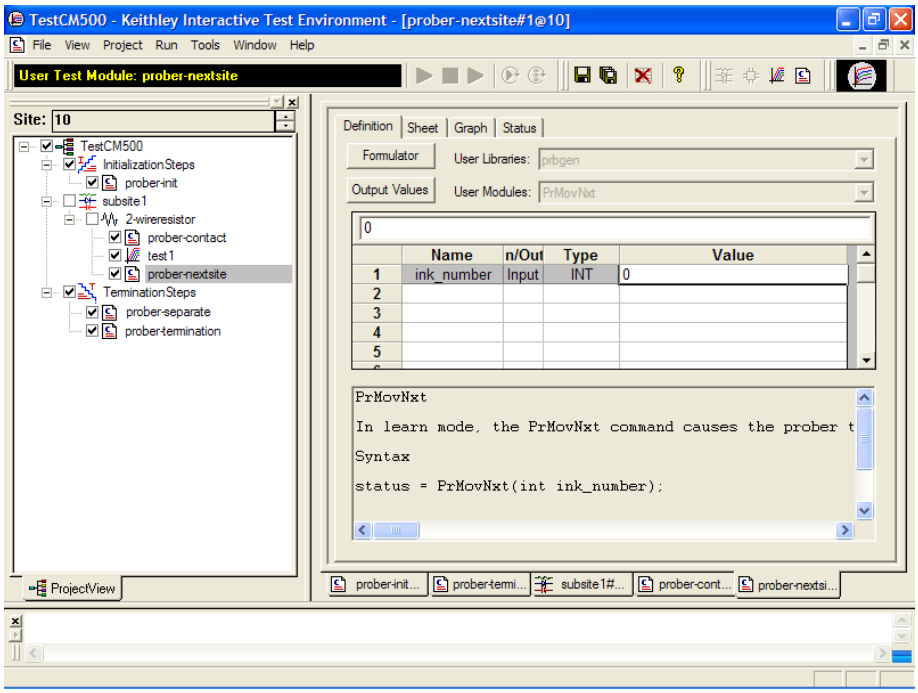
13. Add a new UTM to move prober to next site.
14. Select under User Libraries: **prbgen**.
15. Select under User Modules: **PrMovNxt**.

Figure L-46
Nextsite



16. Set the ink_number to 1 if the user needs to trigger inker1; otherwise, set it to 0.

Figure L-47
Set Ink_number



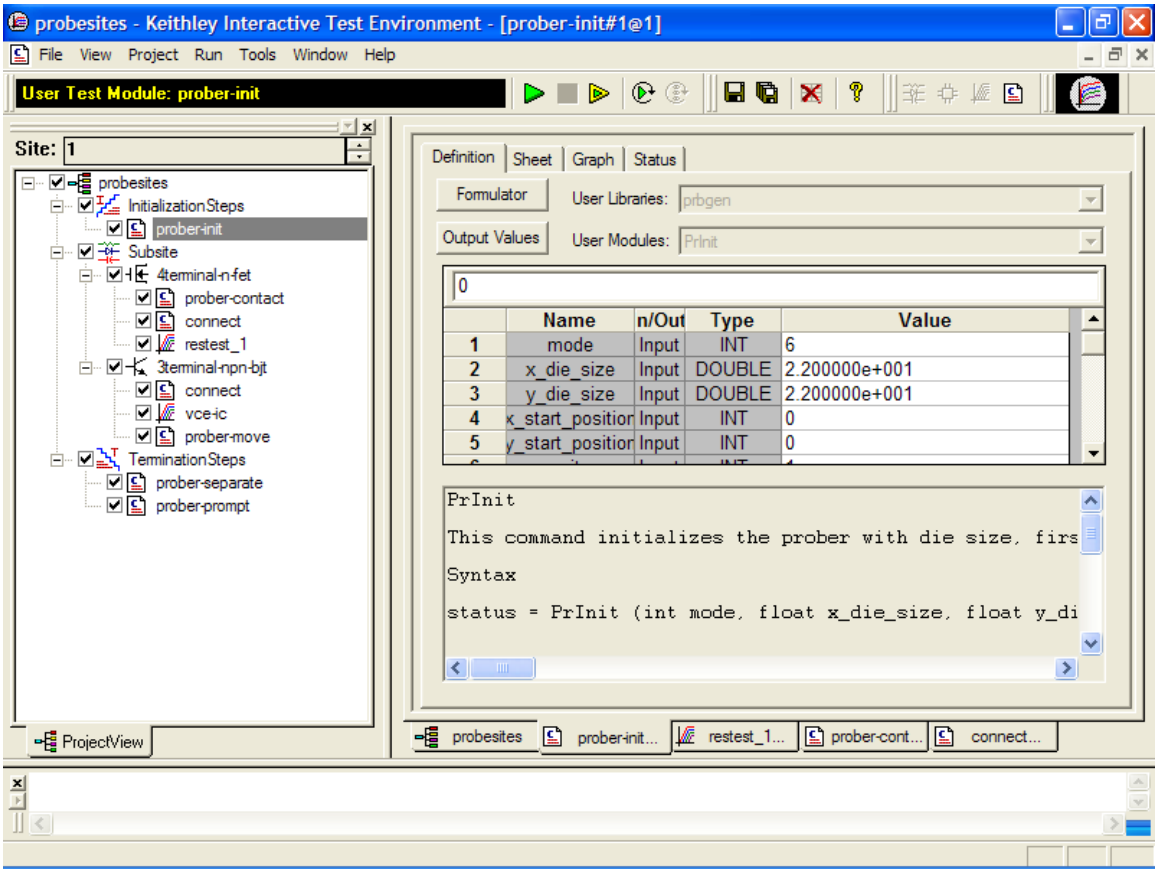
Probesites KITE project example

KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS. Use KITE to open and run the* ***probesites*** *project using the new configuration file, which will allow you to execute the project for this prober.*

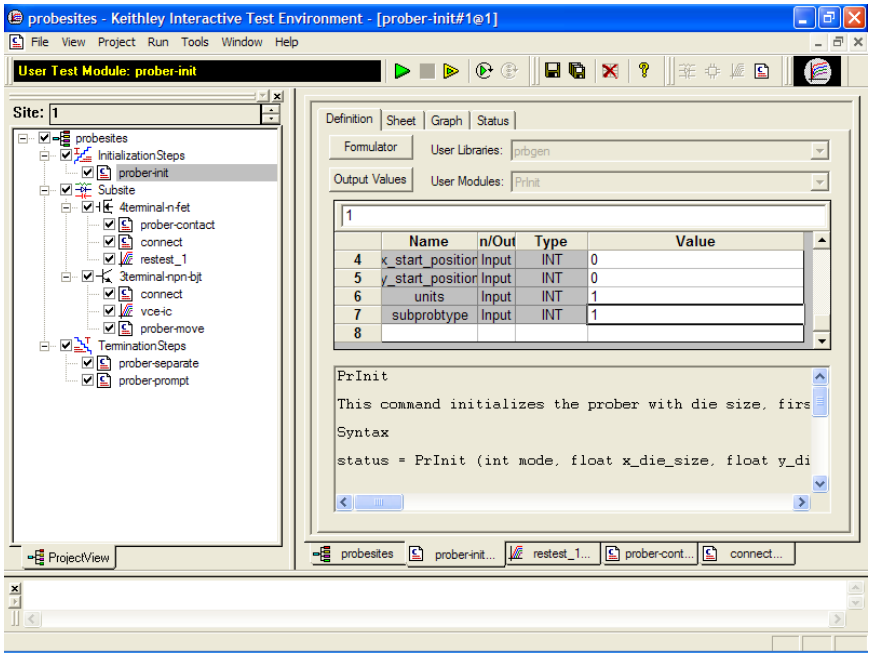
1. Open the **probesites** project example from KITE (Figure G-2).
   a. Select **Open Project** from the **File** menu.
   b. Open the folder: `c:\S4200\kiuser\Projects\probesites`.
   c. Select the project file: **`probesites.kpr`**.
2. Set the first parameter of **PrInit**, which is **mode**, to **6** (see Figure L-48).

Figure L-48
Set PrInit mode parameter



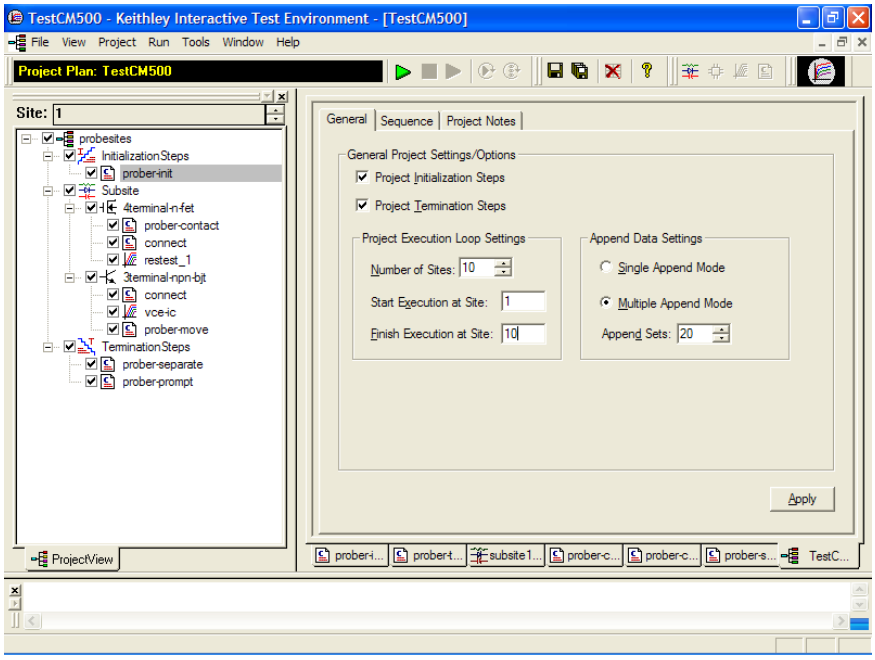
3. Check the last parameter's value. If the CM500 prober is currently not at its first site, set the last parameter subprobtype to 1; otherwise set it to 0.

Figure L-49
Set PrInit subprotype parameter



4. Enter **Project Execution Loop Settings** and click the green **Run** button.

Figure L-50
Run project



Probesubsites KITE project example

The following is a step-by-step procedure to properly configure the KITE project to execute testing and auto wafer stepping to all programmed subsites successfully.

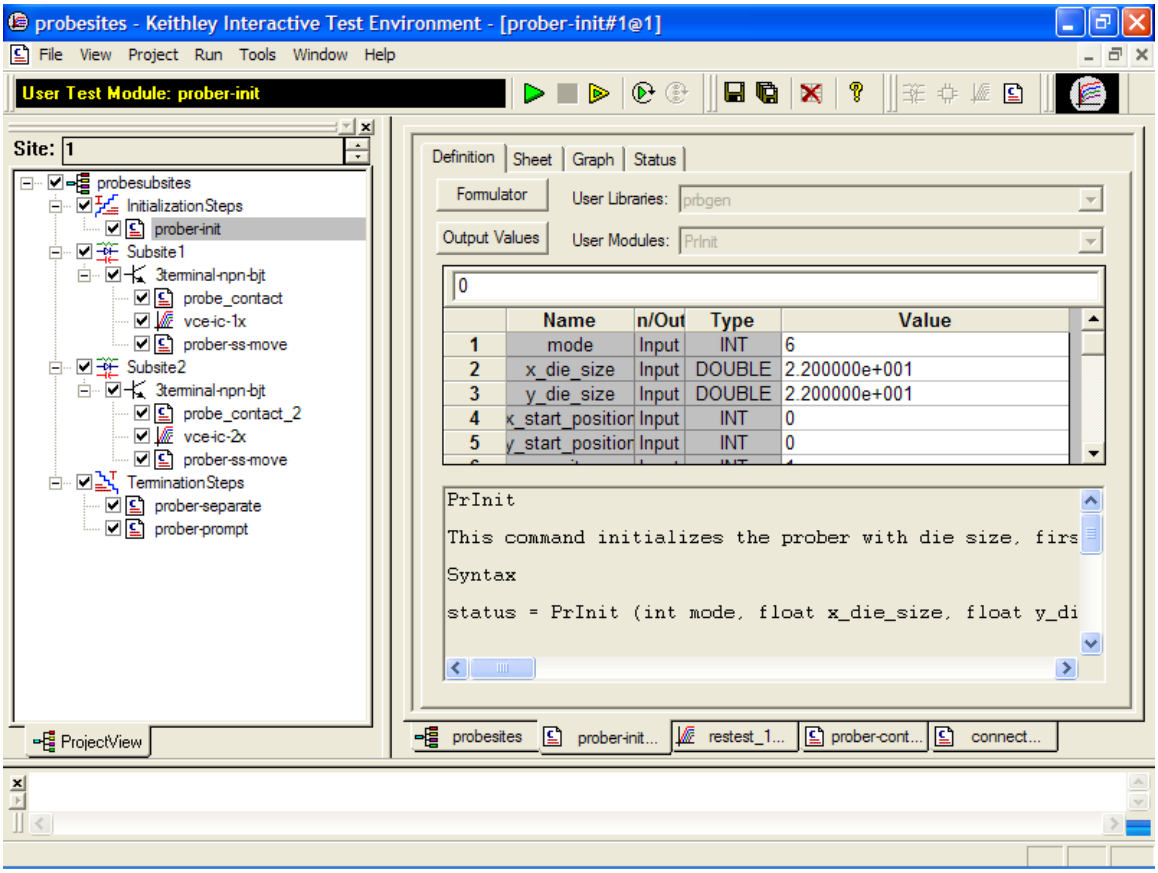
KITE

NOTE: *The following configuration is accomplished using the Model 4200-SCS. Use KITE to open and run the probesubsites project using the new configuration file, which will allow you to execute the project for this prober.*

To open and run the probesubsites project:

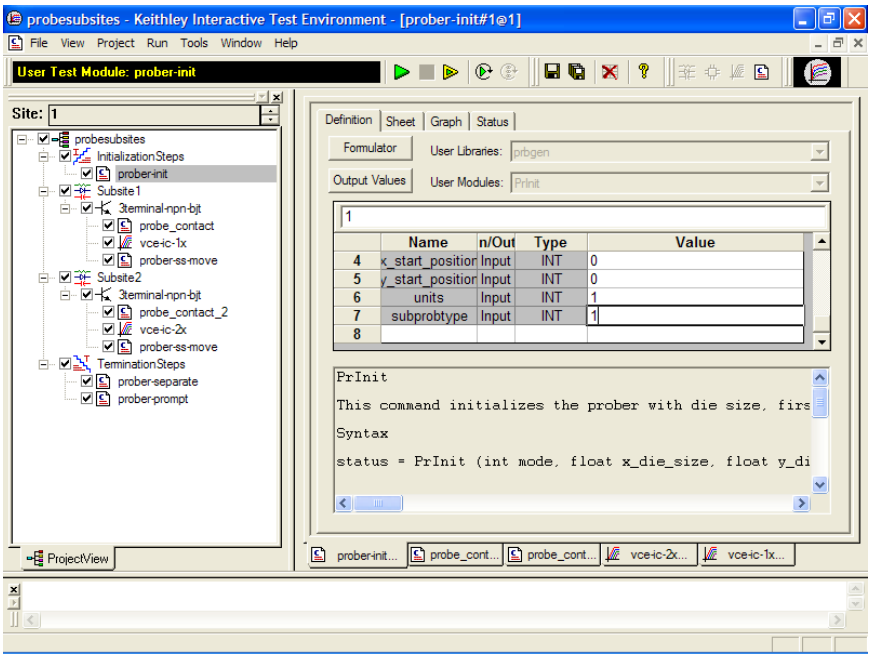
1. Open the **probesubsites** project example from KITE (Figure G-2):
   a. Select **Open Project** from the **File** menu.
   b. Open the folder: `c:\S4200\kiuser\Projects\probesubsites.`
   c. Select the project file: **`probesubsites.kpr.`**
2. Set the first parameter of **PrInit**, which is **mode**, to **6**.

Figure L-51
Set PrInit for probesubsites example



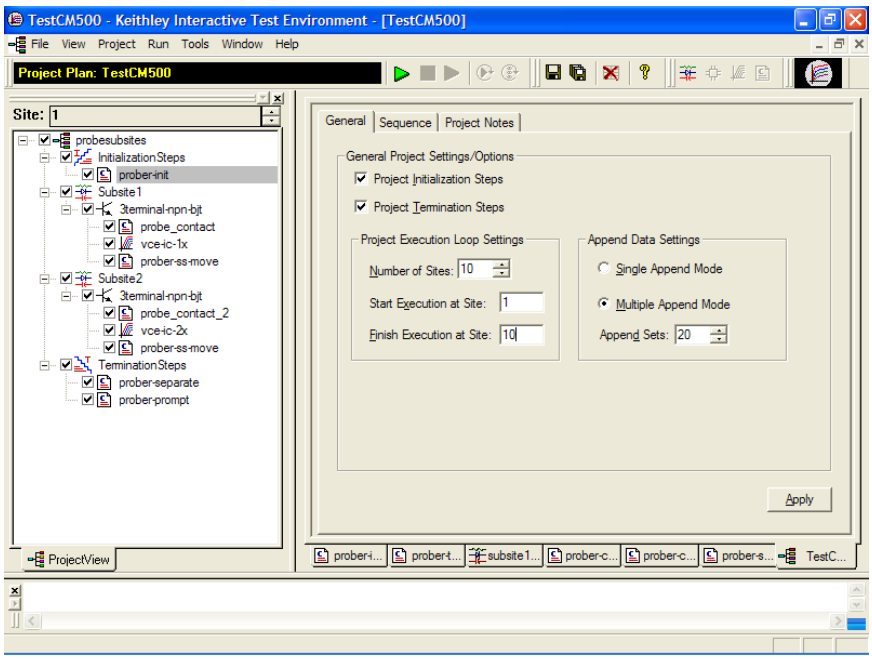
3. Check the last parameter's value. If the CM500 prober is currently not at its first site, set the last parameter subprobtype to 1; otherwise, set it to 0.

Figure L-52
Set subprotype for probesubsites example



4. Enter **Project Execution Loop Settings** and click the green **Run** button.

Figure L-53
Run probesubsites example



Commands and error symbols

The following list (Table L-1) contains error and status symbols listed by command.

Table L-1
Available commands and responses

	PrChuck	PrInit	PrMovNxt	PrSSMovNxt
PR_OK	X	X	X	X
BAD_CHUCK	X			
INVAL_MODE	X			
UNINTEL_RESP	X	X	X	X
BAD_MODE		X		
BAD_MODE		X	X	X
UNEXPE_ERROR		X	X	X
PR_WAFERCOMPLETE			X	X

Appendix M
WLR Testing

In this section:

Introduction

This appendix provides information on WLR testing. Included are tests for HCI, NBTI, Electromigration, and Qbd. Also included is background information on HCI degradation and summaries for using Model 4200-SCS Project Plans to measure HCI degradation and other WLR tests.

NOTE: *The Project Plans for HCI and Qbd testing comply with the standard procedures established by JEDEC. In this appendix, all references to the JEDEC standards and duplicated JEDEC documentation will be clearly indicated as JEDEC copyright protected material.*

JEDEC standards

NOTE: *The following descriptions for the JESD28-A and JESD35-A standard procedures have been acquired from the JEDEC website. This is JEDEC copyright-protected material.*

JESD28-A
Published: Dec-2001

A PROCEDURE FOR MEASURING N-CHANNEL MOSFET HOT-CARRIER-INDUCED DEGRADATION UNDER DC STRESS:

This document describes an accelerated test for measuring the hot-carrier-induced degradation of a single n-channel MOSFET using dc bias. The purpose of this document is to specify a minimum set of measurements so that valid comparisons can be made between different technologies, IC processes, and process variations in a simple, consistent, and controlled way. The measurements specified should be viewed as a starting point in the characterization and benchmarking of the transistor manufacturing process.

JESD35-A
Published: Apr-2001

PROCEDURE FOR WAFER-LEVEL-TESTING OF THIN DIELECTRICS:

The revised JESD35 is intended for use in the MOS Integrated Circuit manufacturing industry. It describes procedures developed for estimating the overall integrity and reliability of thin gate oxides. Three basic test procedures are described: the Voltage-Ramp (V-Ramp), the Current-Ramp (J-Ramp,) and the new Constant Current (Bounded J-Ramp) test. Each test is designed for simplicity, speed, and ease of use. The standard has been updated to include breakdown criteria that are more robust in detecting breakdown in thinner gate oxides that may not experience hard thermal breakdown.

NOTE: *The JEDEC standard procedures are available on the JEDEC website at the following address: http://www.jedec.org*

When you visit the JEDEC website, you must first register before you can access the standards. Registration is free.

HCI degradation: Background information

Hot Carrier Injection (HCI) degradation is one of the most important device issues facing the semiconductor industry. Small gate length and process variations in today's semiconductor process can result in dramatic degradation in HCI device performance. In the last few years HCI lifetimes have reduced dramatically. In some cases, drive current lifetimes have dropped from years to weeks. HCI effects are enhanced with device scaling (this includes a reduction in device gate length). This means that HCI effects will be an even greater concern in the future. HCI is clearly an important semiconductor issue and the need to monitor HCI on a regular basis is a critical test requirement.

Hot carrier damage occurs in MOS devices when carriers (electrons or holes) are accelerated in the channel. In short channel devices, these electrons/holes attain velocities high enough to cause impact ionization. Impact ionization, in turn, creates extra carriers in the MOS channel. These extra carriers result in significant substrate currents and in some cases attain high enough energy to overcome the semiconductor-oxide barrier and are trapped in the oxide. Most of the oxide carrier trapping occurs at the drain edge where carrier velocity is maximized. These trapped channel electrons can cause significant device performance asymmetry and shifts in critical device parameters such as threshold voltage and device drive current. In some cases, as much as 10% change in measured device parameters can occur within a few days.

Today's devices are becoming increasingly susceptible to Hot Carrier effects. In the past, the linear drain current target value for successful hot carrier device performance was a 10% change in 10 years. Typically, today's manufactured devices can no longer meet this specification and as much as 10% degradation in linear drain current can occur in a few days. Because of this fact, the semiconductor manufacturer has even a greater need to monitor HCI effects.

HCI and WLR project plans

There are five Model 4200-SCS project plans for HCI and WLR testing: HCI_1_DUT, HCI_4_DUT, NBTI_1_DUT, EM_const_I, and Qbd.

All but the Qbd project plans use Subsite Cycling in the Stress/Measure Mode. For details, see Subsite cycling in Section 6.

Each of these five projects can be used as configured or modified as needed for your testing requirements.

HCI_1_DUT and HCI_4_DUT project plans

The HCI_1_DUT project plan is shown in Figure M-1. The subsite plan (HC) is configured for subsite cycling using voltage stressing on the single n-channel MOSFET device (4terminal-n-fet). After the first pre-stress cycle to perform characterization tests, subsequent cycles voltage stress the device for a specified period of time before again performing the tests. The Device Stress Properties setup window for the HCI_1_DUT project is shown in Figure M-2.

The HCI_4_DUT project plan is similar to the HCI_1_DUT project plan except it is configured to test four devices using a switching matrix for connections.

In a parallel connection scheme, up to 20 devices can be stressed by voltage. Figure M-3 shows an example of twenty parallel-connected devices being stressed by eight gate and drain voltages.

Figure M-1
HCI_1_DUT project plan

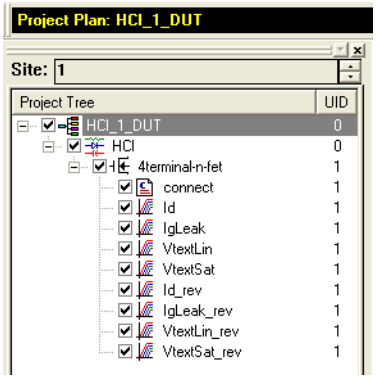


Figure M-2
Device Stress Properties window: HCI_1_DUT project

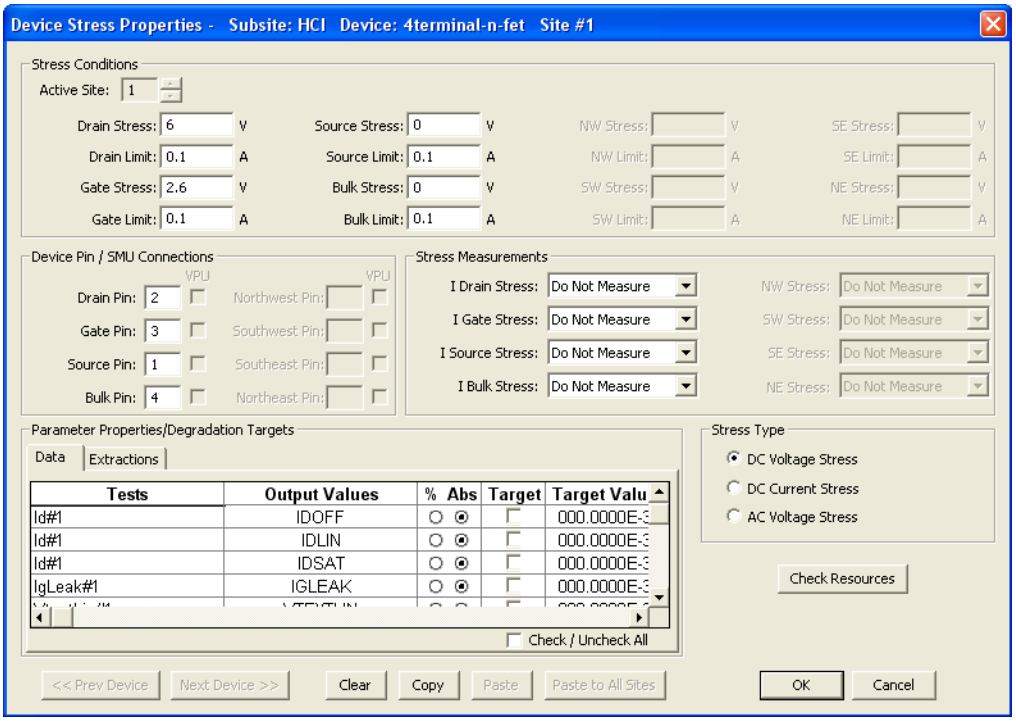
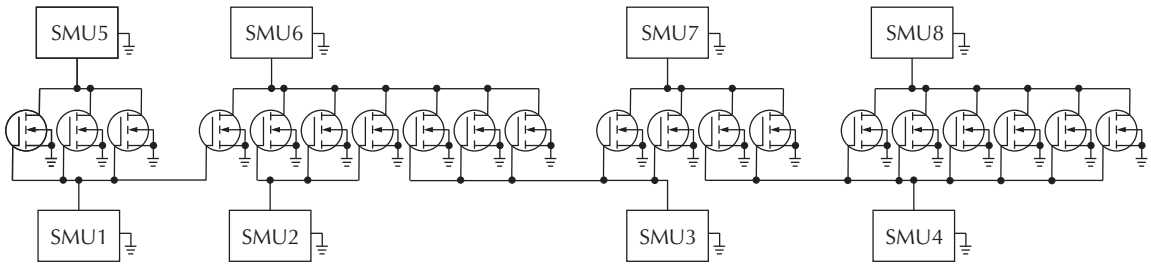


Figure M-3
HCI and NBTI tests: 20 parallel-connected devices stressed by voltage



NBTI_1_DUT project plan

The NBTI_1_DUT project plan is shown in Figure M-4. As with the HCI projects, the subsite plan (NBTI) is configured for subsite cycling using voltage stressing. However, the device is a p-channel MOSFET (PMOS).

This project plan includes initialization and termination steps (UTMs) to control the temperature of the chuck. The subsite plan will not start until the chuck reaches the specified temperature. After the first pre-stress cycle to perform characterization tests, subsequent cycles voltage stress the device for a specified period of time before again performing the tests. The Device Stress Properties setup window for the NBTI_1_DUT project is shown in Figure M-5.

After the subsite plan is completed, the UTM for the termination step cools the chuck.

In a parallel connection scheme, up to 20 devices can be stressed by voltage. Figure M-3 shows an example of twenty parallel-connected devices being stressed by eight gate and drain voltages.

Figure M-4
NBTI_1_DUT project plan

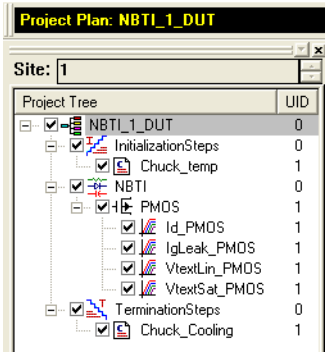
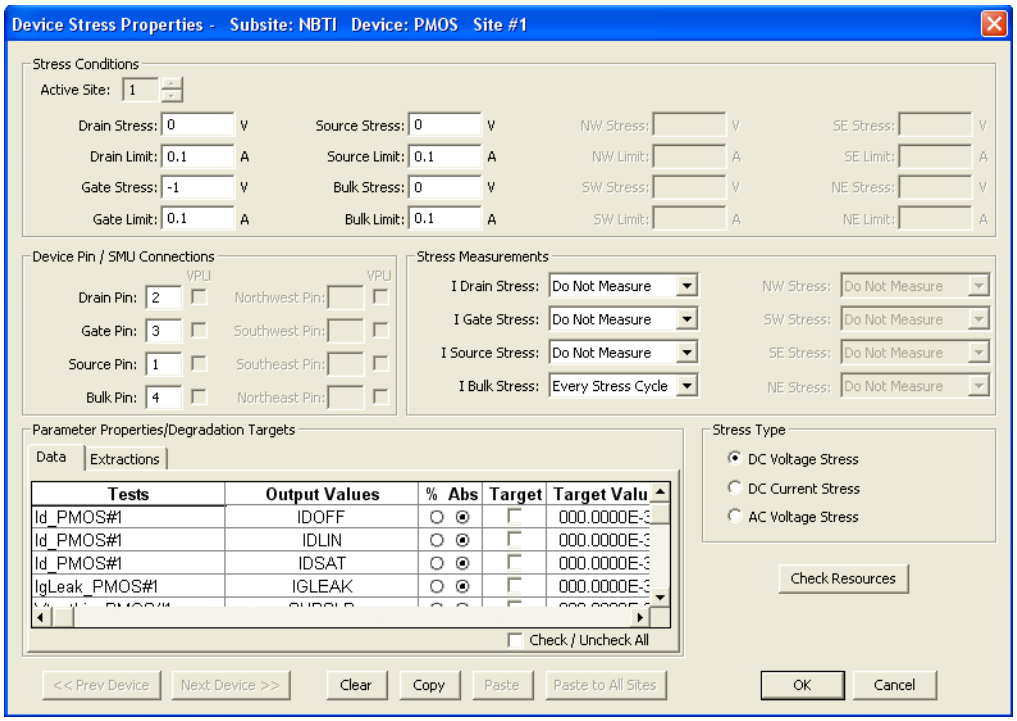


Figure M-5
Device Stress Properties window – NBTI_1_DUT project



EM_const_I project plan

The EM_const_I project plan template is shown in Figure M-6. The subsite plan (EM) is configured for subsite cycling using current stressing on the single device (Metal_Line).

This project plan includes initialization and termination steps (UTMs) to control the temperature of the chuck. The subsite plan will not start until the chuck reaches the specified temperature. After the first pre-stress cycle to perform a characterization test on the device, subsequent cycles current stress the device for a specified period of time before again performing the test. After the subsite plan is completed, the UTM for the termination step cools the chuck. The Device Stress Properties setup window for the EM_const_I project is shown in Figure M-7.

The EM_const_I project plan can be modified to test additional devices. Each SMU in the test system can current-stress one device. Therefore, if there are eight SMUs in the test system, up to eight SMUs can be stressed, as shown in Figure M-8.

NOTE: *Current stressing: When setting the current stress level for each device in the subsite plan, keep in mind that a setting of zero (0) connects the device pin to the ground unit (0 V ground). In order to current stress a device, the current stress level must be set to a non-zero value.*

Figure M-6
EM_const_I project plan

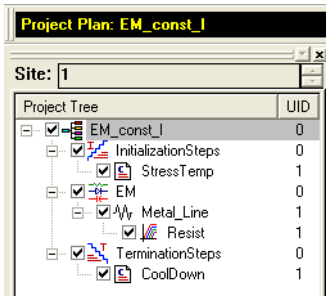


Figure M-7
Device Stress Properties window: EM_const_I project

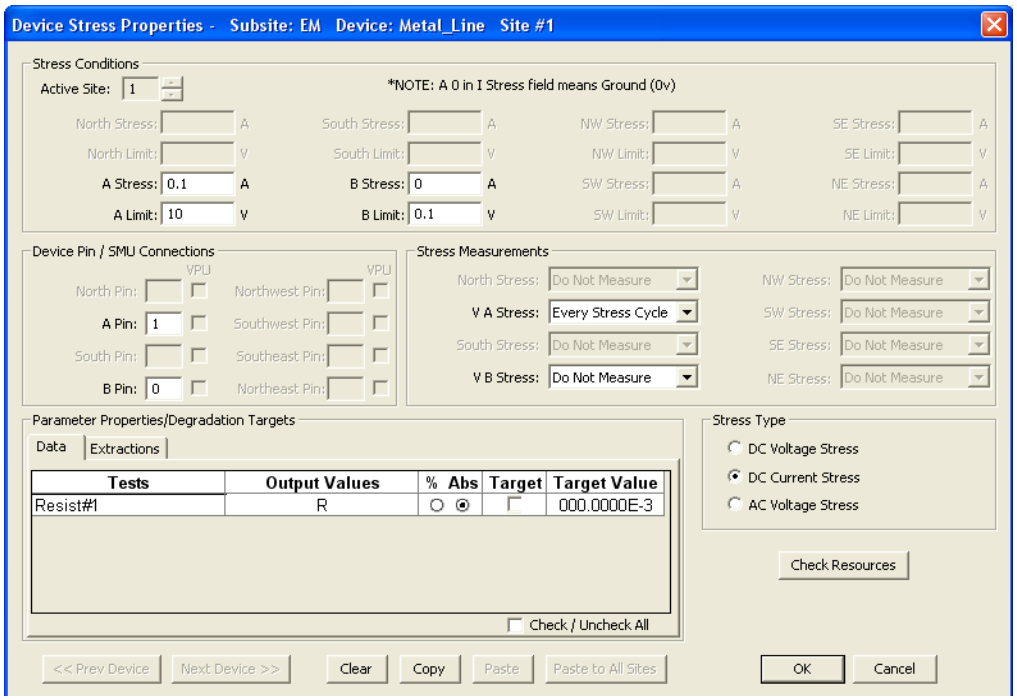
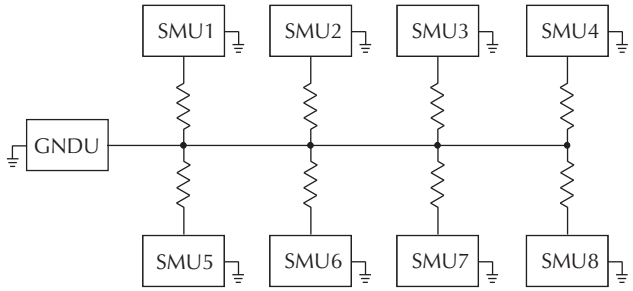


Figure M-8
EM test: Eight devices being current stressed by eight SMUs

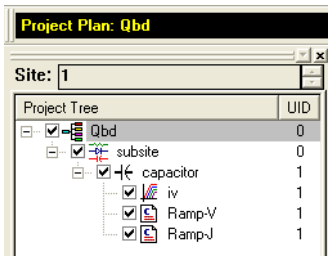


Qbd project plan

The Qbd project plan includes UTMs for the Ramp-V test and the Ramp-J test. These tests adhere to the JESD35-A standard procedures for wafer-level-testing of thin dielectrics. This project (see Figure M-9) does not use subsite cycling.

Details on the these tests, including the User Modules (input and output variables) for the UTMs, are covered in V-ramp and J-ramp tests later in this appendix.

Figure M-9
Qbd project



Configuration sequence for subsite cycling

There are four Project Plans that use subsite cycling. These include HCI_1_DUT, HCI_4_DUT, NBTI_1_DUT, and EM_const_I. The process flow for these projects is shown in Figure M-10.

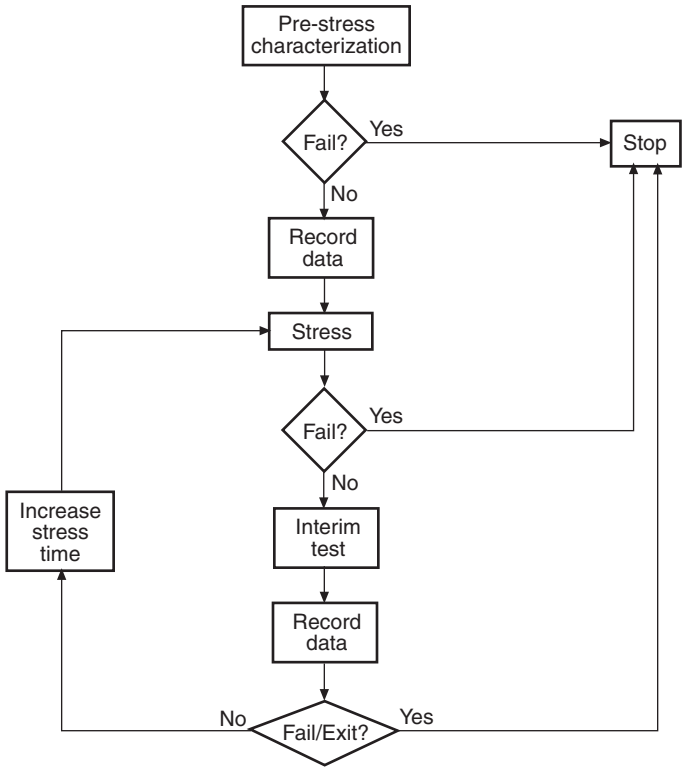
NOTE: *A new project plan for subsite cycling can be created or one of the four existing project plans can be modified as needed. For details, see Building, modifying, and deleting a Project Plan in Section 6.*

When adding a device plan or test to a subsite cycling project, the following sequence must be followed:

1. Insert a Device Plan for the type of device to be tested. For example, if testing a 4-terminal, n-channel MOSFET, insert the 4terminal-n-fet device into the subsite plan.
2. Under the Device Plan, insert a new test (ITM or UTM) or copy a test from the test library and make the proper modifications.
3. Use the **Formulator** for the ITM or UTM to configure data calculations on test data. The window to set the formulator is opened by clicking the **Formulator** button on the Definition tab of the ITM or UTM. For details, see Analyzing test data using the Formulator in Section 6.
4. Select the **Output Values** to be exported to the Subsite Data sheet for inter-stressing monitoring. The window to select Output Values is opened by clicking the **Output Values** button in the **Definition** tab for the ITM or UTM. An Output Value is selected by clicking a checkbox to insert a √. For details, see ITM Output Values and UTM Output Values in Section 6.
5. If desired, **Exit Conditions** for an ITM can be set. When an exit condition other than **None** is selected and the source for the ITM goes into compliance, the test, device plan, subsite plan, site, or project will terminate. **None** is the default exit condition. The window to set the exit condition is opened by clicking the **Exit Conditions** button in the **Definition** tab of the ITM. For details, see ITM compliance exit conditions in Section 6.
6. Save the project plan by selecting **Save All** from the **File** menu (at the top of the KITE window). You can also save the project by clicking the **Save All** button on the toolbar.
7. Repeat steps 2 through 6 for adding more tests for the same device.
8. Repeat steps 1 through 7 for adding more devices to the subsite plan.
9. Configure the subsite for subsite cycling (Stress/Measure Mode: In the Project Navigator, double-click the subsite plan and select the **Subsite Setup** tab to configure subsite cycling.
   a. Set the Stress/Measure Mode cycle times for the subsite plan: The Stress/Measure Mode and cycle times are set from the **Subsite Setup** tab.
   b. Configure the stress properties and connection information for every device in the subsite plan: In the **Subsite Setup** tab, click the **Device Stress Properties** tab to open the properties window.

See Subsite cycling in Section 6. for detailed information on subsite cycling.

Figure M-10
Process flow: HCI/NBTI/constant current EM



V-ramp and J-ramp tests

The V-Ramp test starts at the use-condition voltage (or lower) and ramps linearly from this value until oxide breakdown. The J-Ramp starts at a low current and ramps exponentially until oxide breakdown.

V-ramp test: qbd_rmpv User Module

The V-ramp test is performed by the Ramp-V UTM in the Qbd Project Plan (Figure M-9). This test uses the qbd_rmpv User Module of the wlrlib User Library and is documented as follows:

User Module description

Performs a Charge-to-Breakdown test using the QBD V-ramp test algorithm described in JESD35-A Procedure for Wafer Level Testing of Thin Dielectrics." This algorithm forces a linear voltage ramp until the oxide layer breaks down. This algorithm is capable of a maximum voltage of ±200 volts. The flow diagram for the V-ramp test is shown in Figure M-12.

Figure M-11 shows the default parameters for the qbd_rmpv User Module.

Syntax

status = qbd_rmpv(int hi_pin, int lo_pin1, int lo_pin2, int lo_pin3, char *HiSMUId, char *LoSMUId1, char *LoSMUId2, char *LoSMUId3, double v_use, double I_init, int hold_time, double v_start, double v_step, int t_step, int measure_delay, double I_crit, double I_box, double I_max, double exit_curr_mult, double exit_slope_mult, double q_max, double t_max, double v_max, double area, int exit_mode, double *V_stess, int V_size, double *I_stress, int I_size, double *T_stress, int T_size, double *q_stress, int q_size, double *I_use_pre, double *I_use_post, double *Q_bd, double *q_bd, double *v_bd, double *I_bd, double *t_bd, double *v_crit, double *v_box, int *failure_mode, int *test_status);

Technique

See JEDEC standard JESD35-A PROCEDURE FOR WAFER-LEVEL-TESTING OF THIN DIELECTRICS: at the beginning of this appendix.

NOTE: *Some of the descriptions of the following Input Variables and Output Variables are quoted from the JESD35-A standard. The variables quoted from the standard include this reference identification: (Ref. JESD35-A).*

Figure M-11
qbd_rmpv User Module (default parameters)

Formulator | User Libraries: wlrlib
Output Values | User Modules: qbd_rmpv

	Name	In/Out	Type	Value
1	hi_pin	Input	INT	-1
2	lo_pin1	Input	INT	-1
3	lo_pin2	Input	INT	-1
4	lo_pin3	Input	INT	-1
5	HiSMUId	Input	CHAR_P	SMU1
6	LoSMUId1	Input	CHAR_P	GNDU
7	LoSMUId2	Input	CHAR_P	GNDU
8	LoSMUId3	Input	CHAR_P	GNDU
9	v_use	Input	DOUBLE	-1.000000e+000
10	I_init	Input	DOUBLE	1.000000e-005
11	hold_time	Input	INT	100
12	v_start	Input	DOUBLE	-1.000000e+000
13	v_step	Input	DOUBLE	1.000000e-001
14	t_step	Input	INT	100
15	measure_delay	Input	INT	50
16	I_crit	Input	DOUBLE	1.000000e-005
17	I_box	Input	DOUBLE	1.000000e-004
18	I_max	Input	DOUBLE	5.000000e-002
19	exit_curr_mult	Input	DOUBLE	10
20	exit_slope_mult	Input	DOUBLE	3
21	q_max	Input	DOUBLE	1.000000e+001
22	t_max	Input	DOUBLE	1000
23	v_max	Input	DOUBLE	35
24	area	Input	DOUBLE	2.000000e-004
25	exit_mode	Input	INT	1
26	V_stress	Output	DBL_ARRAY	
27	V_size	Input	INT	1000
28	I_stress	Output	DBL_ARRAY	
29	I_size	Input	INT	1000
30	T_stress	Output	DBL_ARRAY	
31	T_size	Input	INT	1000
32	q_stress	Output	DBL_ARRAY	
33	q_size	Input	INT	1000
34	I_use_pre	Output	DOUBLE_P	
35	I_use_post	Output	DOUBLE_P	
36	Q_bd	Output	DOUBLE_P	
37	q_bd	Output	DOUBLE_P	
38	v_bd	Output	DOUBLE_P	
39	I_bd	Output	DOUBLE_P	
40	t_bd	Output	DOUBLE_P	
41	v_crit	Output	DOUBLE_P	
42	v_box	Output	DOUBLE_P	
43	failure_mode	Output	INT_P	
44	test_status	Output	INT_P	

Input Variables

hi_pin (int) High pin (usually the gate pin) (-1 to 72).

lo_pin1,lo_pin2, lo_pin3 (int) Low pins (enter -1 to NOT connect). low_pin 1, 2, and 3 are usually for source drain and substrate connection. Depending on device structure, some of those pins are optional.

NOTE: *If there is no switching matrix in the system, enter either 0 or -1 for hi_pin and lo_pins to bypass switch.*

HiSMUId (char *) ID string of the SMU outputting the stress.

LoSMUId1, 2, 3 (char *) ID string of the SMU connected to ground terminal. These three IDs can be same.

v_use (double) Oxide voltage (V) under normal operating conditions. Typically the power supply voltage of the process. This voltage is used to measure pre- and post-voltage ramp oxide current (Ref. JESD35-A).

I_init (double) Oxide breakdown failure current when biased at v_use. Typical value is 10 uA/cm^2 and may change depending on oxide area. For maximum sensitivity, the specified value should be well above the worse case oxide current of a "good" oxide and well above the noise level of the measurement system. Higher values must be specified for ultra-thin oxide because of direct tunneling effects (Ref. JESD35-A).

hold_time (init) Time in ms to hold the first stress (v_start).

v_start (double) Starting voltage (V) for voltage ramp. Typical value is v_use (Ref. JESD35-A).

v_step (double) Voltage (V) ramp step height. This value has a maximum value of 0.1 MV/cm. For example, the maximum value can be calculated using Tox*0.1 MV/cm, where Tox is in unit of centimeters. This is 0.1 V for a 10nm oxide (Ref. JESD35-A).

t_step (int) Voltage ramp step time in ms. This is used to determine the voltage ramp rate. This time should be less or equal than 100 ms. Typically 40-100 ms.

measure_delay (int) Time delay in ms for measurement after each voltage stress step. This delay should be less than t_step (ms).

I_crit (double) At least 10 times the test system current measurement noise floor. This oxide current (A) is the minimum value used in determining the change of slope breakdown criteria (Ref. JESD35-A).

I_box (double) An optional measured current level for which a stress voltage is recorded.
This value provides an additional point on the current-voltage curve. A typical value is 1uA (Ref. JESD35-A).

I_max (double) Oxide breakdown criteria. I_bd is obtained from I-V curves and is the oxide current at the step just prior to breakdown (Ref. JESD35-A).

exit_curr_mult (double) Change of current failure criteria. This is the ratio of measured current over previous current level, which, if exceeded, will result in failure (2.5-5, recommended value: 10-100).

exit_slope_mult (double) Change of slope failure criteria. This is the factor of change in FN slope, which, if exceeded, will result in failure (2.5-5, recommended value: 3).

q_max (double) Maximum accumulated oxide charge per oxide area. Used to terminate a test where breakdown occurs but was not detected during the test (C/cm^2) (Ref. JESD35-A).

t_max (double) Maximum stress time allowed in seconds. Reaching this limit will result in test to finish (s).

v_max (double) The maximum voltage limit for the voltage ramp. This limit is specified at 30MV/cm for oxides less than 20nm thick and 15MV/cm for thicker oxides. For example, v_max can be estimated from Tox*30Mv/cm where Tox is in centimeters. This is 35 V for a 10.0nm Oxide (Ref. JESD35-A).

area (double) Area of oxide structure (cm^2).

exit_mode (int) Failure criteria mode:

- **0** Specifies that oxide failure is determined by a measured current that exceeds the user specified failure current (fail_current).
- **1** This mode uses two criteria to determine oxide failure. The first criteria is the specified failure current (fail_current). The second criteria is a slope of current measurement that is a factor (exit_slope_mult) times the previous measured value. See JEDEC document JESD35-A and Addenda (JESD35-1 and JESD35-2).

Because of noise considerations, the calculated failure current criteria is used only when the measured current is 10X the user specified noise current. For measured currents below this value, the fail_current is used as the exit criteria.

V_size, I_size, T_size, q_size Size of data array. Maximum 65535.

Output Variables

***V_stress** (double *) Voltage stress array.

***I_stress** (double *) Measured current array.

***T_stress** (double *) Time stamp array indicating when current is measured.

***q_stress** (double *) Accumulated charge array.

***I_use_pre** (double *) Measured oxide current at v_use, prior to starting the ramp. (Ref. JESD35-A).

***I_use_post** (double *) Measured oxide current at v_use, after the ramp finished. (Ref. JESD35-A).

***Q_bd** (double *) Charge-to-breakdown. Cumulative charge passing through the oxide prior to breakdown (C). (Ref. JESD35-A).

***q_bd** (double *) Charge-to-breakdown density (C/cm^2). (Ref. JESD35-A).

***v_bd** (double *) Applied voltage at the step just before oxide breakdown. (Ref. JESD35-A).

***I_bd** (double *) Measured current at v_bd, just before oxide breakdown.

***t_bd** (double *) Time stamp when measuring I_bd.

***v_cri** (double *) Applied voltage at the step when the oxide current exceeds I_crit. (Ref. JESD35-A).

***v_box** (double *) Applied voltage at the step when the oxide current exceeds I_box. (Ref. JESD35-A).

***failure_mode** (int *)

- **1** Initial test failure.
- **2** Catastrophic failure (initial test pass, ramp test fail, post test fail).

3 Masked Catastrophic (initial test pass, ramp test pass, post test fail).

4 Non-Catastrophic (initial test pass, ramp test fail, post test pass).

5 Others (initial test pass, ramp test pass, post test pass).

***test_status (int *)**

2 No test errors (exit due to measured current > a factor of the previous measurement).

1 No test errors (exit due to measured current slope > a factor of the previous slope).

0 No test errors (exit due to measured current > fail_current ONLY).

-1 Failed pre-stress test.

-2 Cumulative charge limit reached.

-3 Voltage limit reached.

-4 Maximum time limit reached.

-5 Masked Catastrophic Failure.

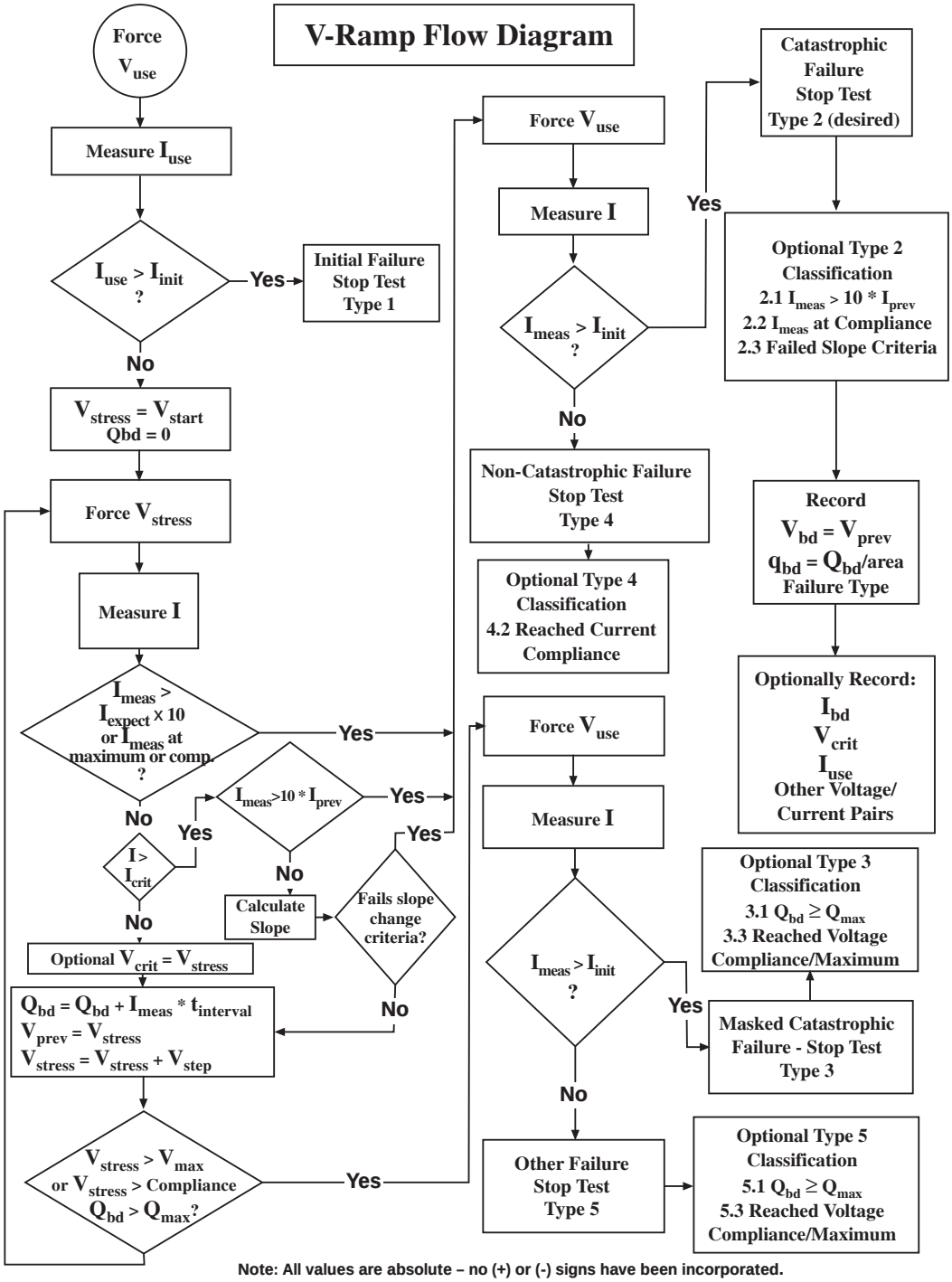
-6 Non-Catastrophic Failure.

-7 Invalid specified t_step, hold_time, or measure_delay.

NOTE: *Invalid Test Result - Result = 1e21.*

Figure M-12
Detailed V-ramp flow diagram

NOTE: *The following diagram from JESD35-A has been reproduced with permission from JEDEC. This flowchart is JEDEC*



Note: All values are absolute – no (+) or (-) signs have been incorporated.

J-ramp test: qbd_rmpj User Module

The J-ramp test is performed by the Ramp-J UTM in the Qbd Project Plan, which is shown in Figure M-9. This test uses the qbd_rmpj User Module of the wlrlib User Library and is documented as follows:

User Module description

Performs a Charge-to-Breakdown test using the QBD J-ramp test algorithm described in JESD35-A "Procedure for Wafer Level Testing of Thin Dielectrics." This algorithm forces a logarithmic current ramp until the oxide layer breaks down. This algorithm is capable of a maximum current of +/- 1A if a high power SMU is used. The flow diagram for the V-ramp test is shown in Figure M-14.

Figure M-13 shows the default parameters for the qbd_rmpj User Module.

Syntax

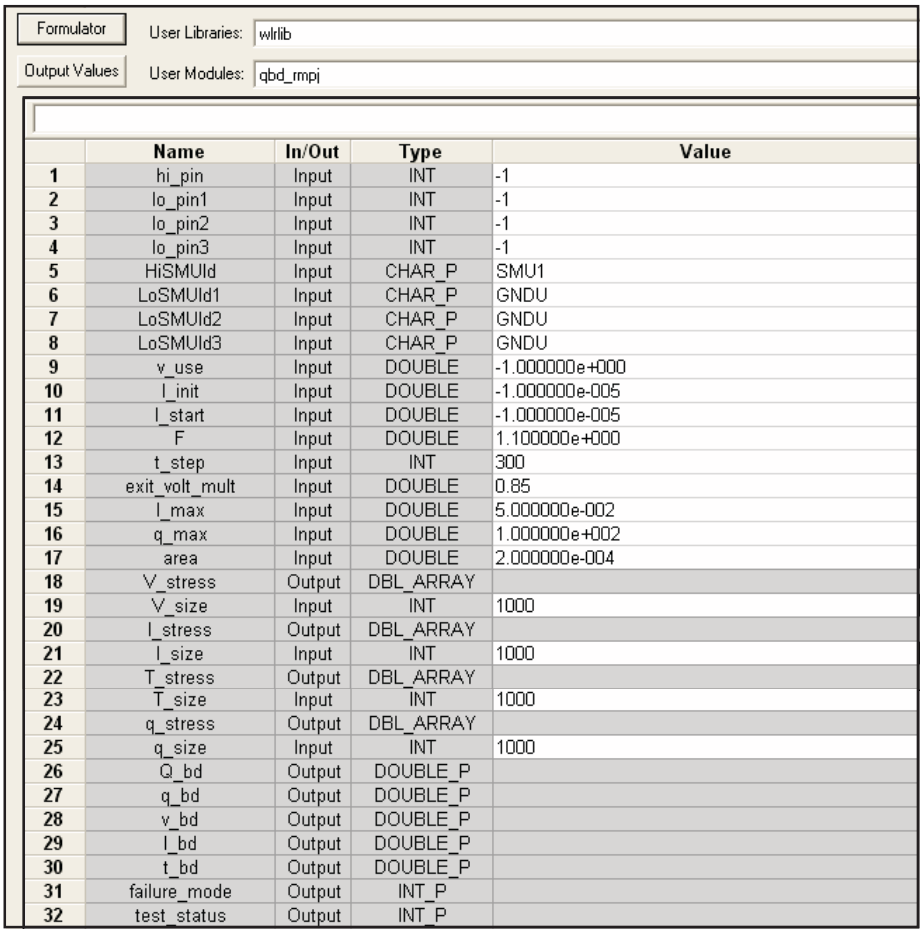
status = qbd_rmpj(int hi_pin, int lo_pin1, int lo_pin2, int lo_pin3, char *HiSMUId, char *LoSMUId1, char *LoSMUId2, char *LoSMUId3, double v_use, double I_init, double I_start, double F, int t_step, double exit_volt_mult, double I_max, double q_max, double area, double *V_stress, int V_size, double *I_stress, int I_size, double *T_stress, int T_size, double *q_stress, int q_size, double *Q_bd, double *q_bd, double *v_bd, double *I_bd, double *t_bd, int *failure_mode, int *test_status)

Technique

See JEDEC standard JESD35-A "Procedure for Wafer-Level-Testing of Thin Dielectrics," referenced in Appendix L.

NOTE: *Some of the descriptions of the following Input Variables and Output Variables are quoted from the JESD35-A standard. The variables quoted from the standard include this reference identification: (Ref. JESD35-A).*

Figure M-13
qbd_rmpj User Module (default parameters)



	Name	In/Out	Type	Value
1	hi_pin	Input	INT	-1
2	lo_pin1	Input	INT	-1
3	lo_pin2	Input	INT	-1
4	lo_pin3	Input	INT	-1
5	HiSMUId	Input	CHAR_P	SMU1
6	LoSMUId1	Input	CHAR_P	GNDU
7	LoSMUId2	Input	CHAR_P	GNDU
8	LoSMUId3	Input	CHAR_P	GNDU
9	v_use	Input	DOUBLE	-1.000000e+000
10	I_init	Input	DOUBLE	-1.000000e-005
11	I_start	Input	DOUBLE	-1.000000e-005
12	F	Input	DOUBLE	1.100000e+000
13	t_step	Input	INT	300
14	exit_volt_mult	Input	DOUBLE	0.85
15	I_max	Input	DOUBLE	5.000000e-002
16	q_max	Input	DOUBLE	1.000000e+002
17	area	Input	DOUBLE	2.000000e-004
18	V_stress	Output	DBL_ARRAY	
19	V_size	Input	INT	1000
20	I_stress	Output	DBL_ARRAY	
21	I_size	Input	INT	1000
22	T_stress	Output	DBL_ARRAY	
23	T_size	Input	INT	1000
24	q_stress	Output	DBL_ARRAY	
25	q_size	Input	INT	1000
26	Q_bd	Output	DOUBLE_P	
27	q_bd	Output	DOUBLE_P	
28	v_bd	Output	DOUBLE_P	
29	I_bd	Output	DOUBLE_P	
30	t_bd	Output	DOUBLE_P	
31	failure_mode	Output	INT_P	
32	test_status	Output	INT_P	

Input Variables

hi_pin (int) High pin (usually the gate pin) (-1 to 72).

lo_pin1,lo_pin2, lo_pin3 (int) Low pins (enter -1 to NOT connect). low_pin 1, 2, and 3 are usually for source drain and substrate connection. Depending on device structure, some of those pins are optional.

NOTE: *If there is no switching matrix in the system, input either 0 or -1 for hi_pin and lo_pins to bypass switch.*

HiSMUId (char *) ID string of the SMU outputting stress.

loSMUId1, 2, 3 (char *) ID string of the SMU connected to ground terminal. These three IDs can be same.

v_use (double) Oxide voltage (V) under normal operating conditions. Typically the power supply voltage of the process. This voltage is used to measure pre- and post-voltage ramp oxide current (Ref. JESD35-A).

I_init (double) Oxide breakdown failure current when biased at v_use. Typical value is 10 uA/cm^2 and may change depending on oxide area. For maximum sensitivity, the specified value should be well above the worse case oxide current of a "good" oxide and well above the system noise

	floor. Higher values must be specified for ultra-thin oxide because of direct tunneling effects (Ref. JESD35-A).
I_start	(double) Starting current (A) for current ramp. Typical value is I_init (Ref. JESD35-A).
F	(double) Current multiplier between two successive current steps (Ref. JESD35-A).
t_step	(init) Current ramp step time (s) (Ref. JESD35-A).
exit_volt_mult	(double) Multiplier factor of successive voltage measurements. When the next measured voltage is below this factor multiplying the previous measured voltage, oxide is considered to be at breakdown and the test will exit. Typical value 0.85.
I_max	(double) Maximum ramp current (A) (Ref. JESD35-A).
q_max	(double) Maximum accumulated oxide charge per oxide area. Used to terminate a test where breakdown occurs but was not detected during the test (C/cm^2) (Ref. JESD35-A).
area	(double) area of oxide structure (cm^2).
V_size, I_size, T_size, q_size	(int) Size of data array. Maximum 65535.

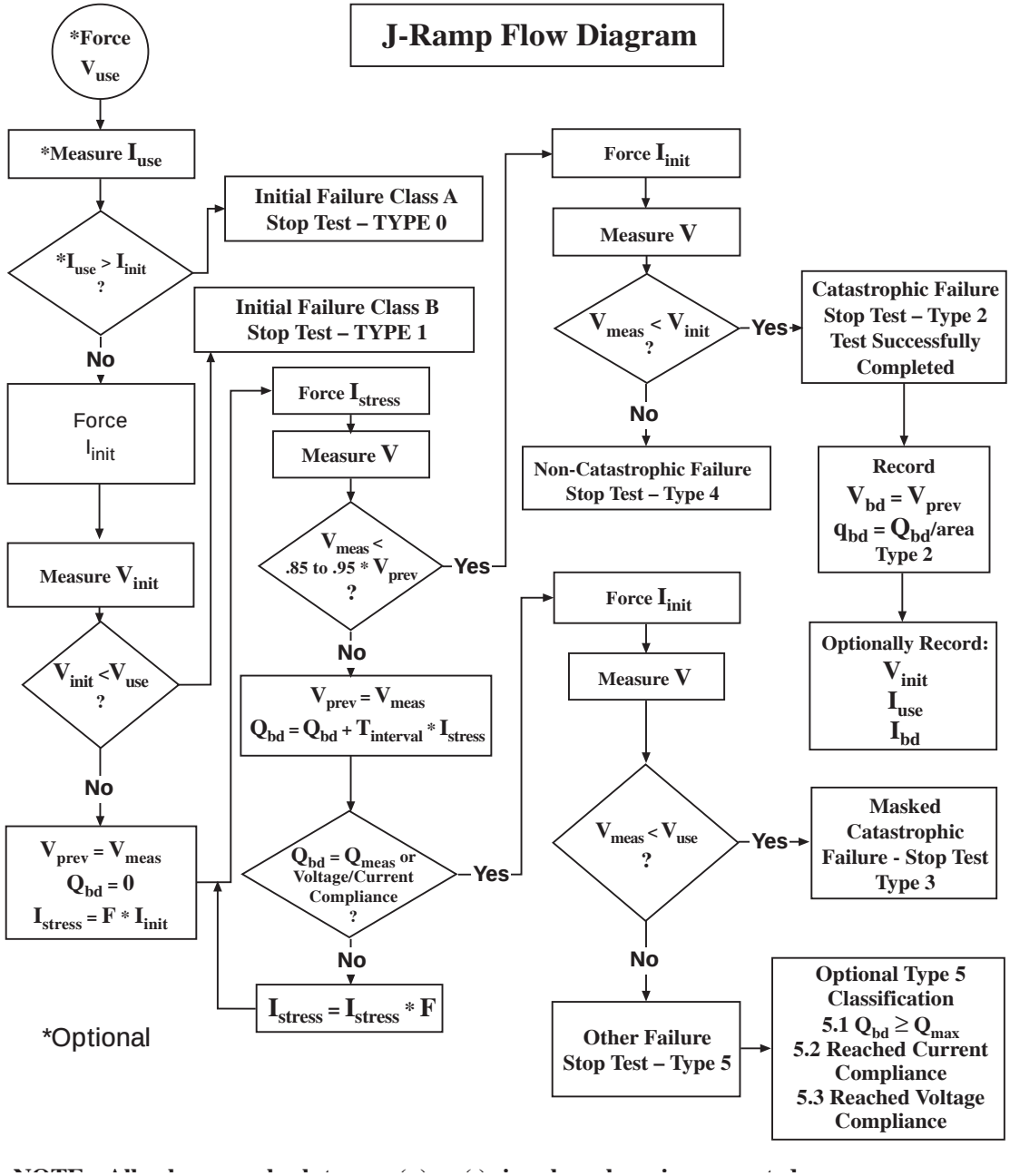
Output variables:

***V_stress**	(double *) Voltage stress array.
***I_stress**	(double *) Measured current array.
***T_stress**	(double *) Time stamp array indicating when current is measured.
***q_stress**	(double *) Accumulated charge array.
***Q_bd**	(double *) Charge-to-breakdown. Cumulative charge (C) passing through the oxide prior to breakdown (Ref. JESD35-A).
***q_bd**	(double *) Charge-to-breakdown density (C/cm^2) (Ref. JESD35-A).
***v_bd**	(double *) Applied voltage at the step just before oxide breakdown (Ref. JESD35-A).
***I_bd**	(double *) Measured current at v_bd, just before oxide breakdown.
***t_bd**	(double *) Time stamp when measuring I_bd.
***failure_mode** (int *)	
1	Initial test failure.
2	Catastrophic failure (initial test pass, ramp test fail, post test fail).
3	Masked Catastrophic (initial test pass, ramp test pass, post test fail).
4	Non-Catastrophic (initial test pass, ramp test fail, post test pass).
5	Others (initial test pass, ramp test pass, post test pass).
***test_status** (int *)	
0	No errors (exit due to measured voltage < factor of the previous value).
-1	Failed pre-stress test.
-2	Cumulative charge limit reached.
-3	Maximum time limit reached.
-4	Masked Catastrophic Failure.
-5	Non-Catastrophic Failure.
-6	Invalid specified t_step.

NOTE: *Invalid Test Result - Result = 1e21.*

Figure M-14
Detailed J-ramp flow diagram

NOTE: *The following diagram from JESD35-A has been reproduced with permission from JEDEC. This flowchart is JEDEC copyright protected material.*



NOTE: *All values are absolute; no (+) or (-) signs have been incorporated.*

Appendix N
Additional User Libraries

In this section:

hp4294ulib user library reference

The user modules in the hp4294ulib user library are used to calibrate and control the HP Model 4294 IMP meter. Subroutines are provided to perform voltage or frequency sweeps. These user modules are summarized in Table N-1.

Details for each of the user modules follow the table.

Table N-1
HP Model 4294 IMP user modules

User module	Description
CvSweep4294	Performs a capacitance vs. voltage sweep.
FiSweep4294	Performs a frequency vs. impedance sweep.
LoadCal4294	Performs LOAD calibration.
OpenCal4294	Performs OPEN calibration.
PhaseCal4294	Performs PHASE calibration.
ShortCal4294	Performs SHORT calibration.

An HP 4294 measurement is valid only if proper calibrations are performed prior to it. The user may run calibration at any time.

A recommended calibration sequence is as follows:

1. Move prober to an OPEN calibration structure.
2. Call `PhaseCal4294`.
3. Call `OpenCal4294`.
4. Move prober to a SHORT calibration structure.
5. Call `ShortCal4294`.
6. Move prober to a LOAD calibration structure.
7. Call `LoadCal4294`.

NOTE: *The HP 4294 is added to the Model 4200-SCS test system using KCON. For details, see Keithley CONfiguration Utility (KCON) in Section 7.*

NOTE: *Details on HP 4294 operations are provided in the documentation provided by HP for the IMP meter.*

CvSweep4294 user module

User module description

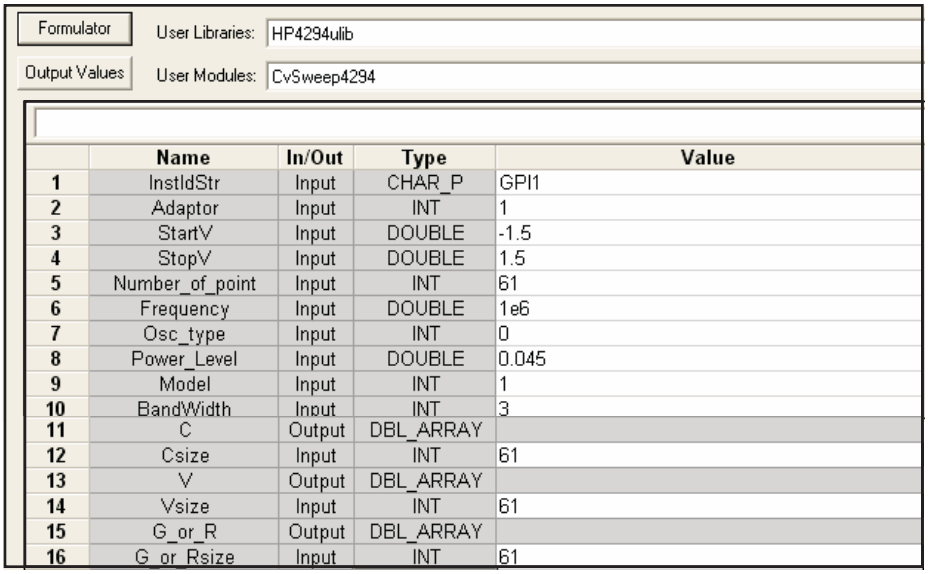
The CvSweep4294 routine performs a capacitance vs. voltage (C-V) sweep using the HP 4294 IMP meter. Figure N-1 shows the default parameters for the CvSweep4294 user module. In general, the HP 4294 outputs a linear staircase voltage sweep. The shape of the staircase is configured by user inputs for the starting voltage, ending voltage, and number of points in the sweep. A capacitance measurement is performed on each step of the sweep.

Syntax:

status = CvSweep4294(char *Inststr, int Adaptor, double StartV, double StopV, double Frequency, int Osc_type, double Power_Level, int Model, int Bandwidth,

double V[], int Number_of_point, double C[], int Csize, double G_or_R[], int G_or_Rsize);

Figure N-1
CvSweep4294 user module (default parameters)



	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	GPI1
2	Adaptor	Input	INT	1
3	StartV	Input	DOUBLE	-1.5
4	StopV	Input	DOUBLE	1.5
5	Number_of_point	Input	INT	61
6	Frequency	Input	DOUBLE	1e6
7	Osc_type	Input	INT	0
8	Power_Level	Input	DOUBLE	0.045
9	Model	Input	INT	1
10	BandWidth	Input	INT	3
11	C	Output	DBL_ARRAY	
12	Csize	Input	INT	61
13	V	Output	DBL_ARRAY	
14	Vsize	Input	INT	61
15	G_or_R	Output	DBL_ARRAY	
16	G_or_Rsize	Input	INT	61

INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

1 Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
2 Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
3 Specifies Agilent 42942A (7-millimeter 42942A probe).
4 Specifies Agilent 42941A (Advanced impedance probe).

StartV (double) Starting voltage of the sweep. Valid range of inputs is -40 to 40 V.

StopV (double) Ending voltage of the sweep. Valid range of inputs is -40 to 40 V.

Number_of_point (int) Number of points in the voltage sweep. Valid range of inputs is 8 to 801 points.

Frequency (double) Measurement frequency of the sweep. Valid inputs are 20 through 1.1 E+8 Hz.

Osc_type (int) Type of AC stimulus: 0 = voltage, 1 = current. Default is voltage.

Power_Level (double) OSC source level:

Voltage: Valid inputs are 5 E-3 to 1 volt. Minimum step is 1 E-3 volts. Default is 5 E-1 volts.

Current: Valid inputs are 2 E-8 to 2 E-4 amps. Minimum step is 1 E-7 amps. Default is 2 E-8 volts.

Model (int) Measurement model. Supported models include the following:

CsRs

CpG

R-X

G-B

Z-Theta

CpD

BandWidth (int) Measurement bandwidth factor. Valid inputs are from 1 to 5.

Csize, Vsize, G or R Size (int) Must be equal to or greater than the specified Number_of_point.

OUTPUTS:

C (double *) Array that stores the capacitance values.

V (double *) Array that stores the bias voltage.

G_or_R (double *) Array that stores the conductance (G) or resistance (R) values.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0 OK.

-**10030** HP 4294 not in KCON.

-**10090** GPIB time-out occurred during communications.

-**10100** An invalid input parameter is specified.

-**10102** There is an error when parsing the HP 4294's response.

FiSweep4294 user module

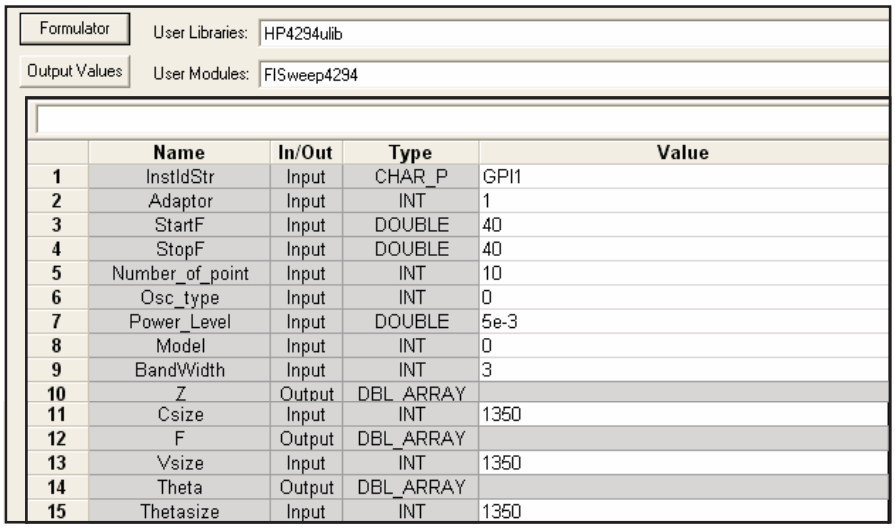
User module description

The FiSweep4294 routine performs a frequency vs. impedance sweep (C-Z) using the HP 4294 IMP meter. Figure N-2 shows the default parameters for the FiSweep4294 user module. In general, the HP 4294 outputs a linear staircase frequency sweep. The shape of the staircase is configured by user inputs for the starting frequency, ending frequency, and number of points in the sweep. An impedance measurement is performed on each step of the sweep.

Syntax:

status = FiSweep4294(char *Inststr, int Adaptor, double StartF, double StopF, int Osc_type, double Power_Level, int Model, int Bandwidth, double V[], int Number_of_point, double C[], int Csize, double G_or_R[], int G_or_Rsize);

Figure N-2
FiSweep4294 user module (default parameters)



	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	GPI1
2	Adaptor	Input	INT	1
3	StartF	Input	DOUBLE	40
4	StopF	Input	DOUBLE	40
5	Number_of_point	Input	INT	10
6	Osc_type	Input	INT	0
7	Power_Level	Input	DOUBLE	5e-3
8	Model	Input	INT	0
9	BandWidth	Input	INT	3
10	Z	Output	DBL_ARRAY	
11	Csize	Input	INT	1350
12	F	Output	DBL_ARRAY	
13	Vsize	Input	INT	1350
14	Theta	Output	DBL_ARRAY	
15	Thetasize	Input	INT	1350

INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

1 Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
2 Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
3 Specifies Agilent 42942A (7-millimeter 42942A probe).
4 Specifies Agilent 42941A (Advanced impedance probe).

StartF (double) Starting frequency of the sweep. Valid range of inputs is 40 to 1.1E+8 Hz volts.

StopF (double) Ending frequency of the sweep. Valid range of inputs is 40 to 1.1E+8 Hz volts.

Number_of_point (int) Number of points in the voltage sweep. Valid range of inputs is 8 to 801 points.

Osc_type (int) Type of AC stimulus: 0 = voltage, 1 = current. Default is voltage.

Power_Level (double) OSC source level. Valid inputs are 0.005 to 1 volt.

Model (int) Measurement model. Supported models include the following:

CsRs

CpG

R-X

G-B

Z-Theta

CpD

BandWidth (int) Measurement bandwidth factor. Valid inputs are from 1 to 5.

Csize, Vsize, Thetasize (int) Must be equal to or greater than the specified Number_of_point.

OUTPUTS:

Z	(double *) Array that stores the impedance values.
F	(double *) Array that stores the frequency points.
Theta	(double *) Array that stores theta values.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0	OK.
-10030	HP 4294 not in KCON.
-10090	GPIB time-out occurred during communications.
-10100	An invalid input parameter is specified.
-10102	There is an error when parsing the HP 4294's response.

LoadCal4294 user module

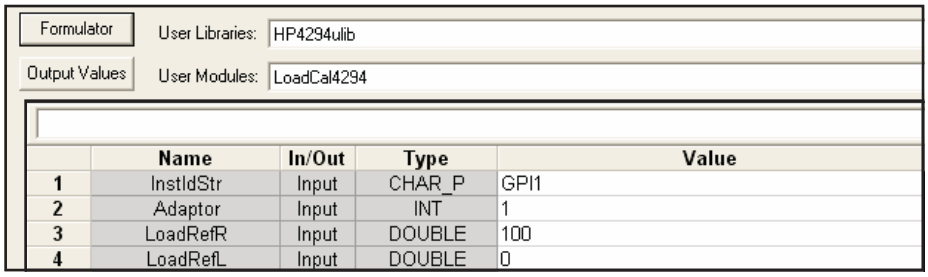
User module description

The LoadCal4294 routine performs LOAD calibration for the HP 4294 IMP meter. Figure N-3 shows the default parameters for the LoadCal4294 user module. After LOAD calibration is performed, it should then be verified by performing an FiSweep on the calibration device.

Syntax:

status = LoadCal4294(char *Inststr, int Adaptor, double LoadRefR, double LoadRefL);

Figure N-3
LoadCal4294 user module (default parameters)



INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

- **1** Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
- **2** Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
- **3** Specifies Agilent 42942A (7-millimeter 42942A probe).
- **4** Specifies Agilent 42941A (Advanced impedance probe).

LoadRefR (double) Reference resistance value of the LOAD.

LoadRefL (double) Reference inductance value of the LOAD.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0	OK.
-**10030**	HP 4294 not in KCON.
-**10090**	GPIB time-out occurred during communications.
-**10100**	An invalid input parameter is specified.
-**10102**	There is an error when parsing the HP 4294's response.

OpenCal4294 user module

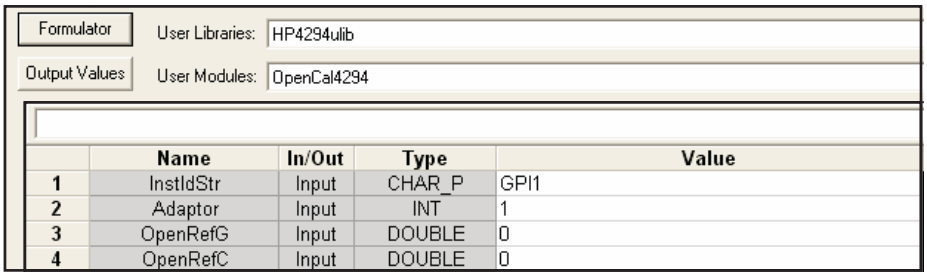
User module description

The OpenCal4294 routine performs OPEN calibration for the HP 4294 IMP meter. Figure N-4 shows the default parameters for the OpenCal4294 user module. After OPEN calibration is performed, it should then be verified by performing an FiSweep on the calibration device.

Syntax:

status = OpenCal4294(char *Inststr, int Adaptor, double OpenRefG, double OpenRefC);

Figure N-4
OpenCal4294 user module (default parameters)



	Name	In/Out	Type	Value
1	InstIdStr	Input	CHAR_P	GPI1
2	Adaptor	Input	INT	1
3	OpenRefG	Input	DOUBLE	0
4	OpenRefC	Input	DOUBLE	0

INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

- **1** Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
- **2** Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
- **3** Specifies Agilent 42942A (7-millimeter 42942A probe).
- **4** Specifies Agilent 42941A (Advanced impedance probe).

OpenRefG (double) Reference conductance value of the OPEN.

OpenRefC (double) Reference capacitance value of the OPEN.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0	OK.
-**10030**	HP 4294 not in KCON.
-**10090**	GPIB time-out occurred during communications.

-10100 An invalid input parameter is specified.

-10102 There is an error when parsing the HP 4294's response.

PhaseCal4294 user module

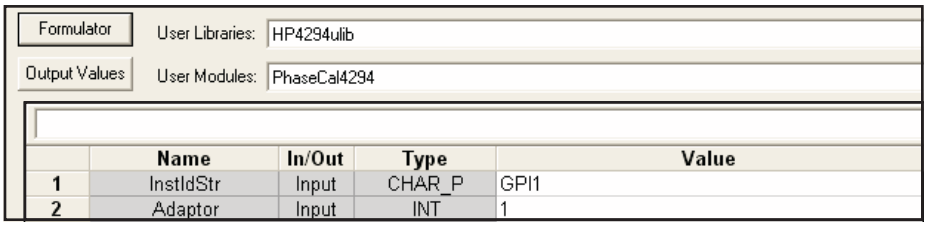
User module description

The PhaseCal4294 routine performs PHASE calibration for the HP 4294 IMP meter. Figure N-5 shows the default parameters for the PhaseCal4294 user module.

Syntax:

status = PhaseCal4294(char *Inststr, int Adaptor);

Figure N-5
PhaseCal4294 user module (default parameters)



INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

1 Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
2 Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
3 Specifies Agilent 42942A (7-millimeter 42942A probe).
4 Specifies Agilent 42941A (Advanced impedance probe).

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0 OK.

-10030 HP 4294 not in KCON.

-10090 GPIB time-out occurred during communications.

-10100 An invalid input parameter is specified.

-10102 There is an error when parsing the HP 4294's response.

ShortCal4294 user module

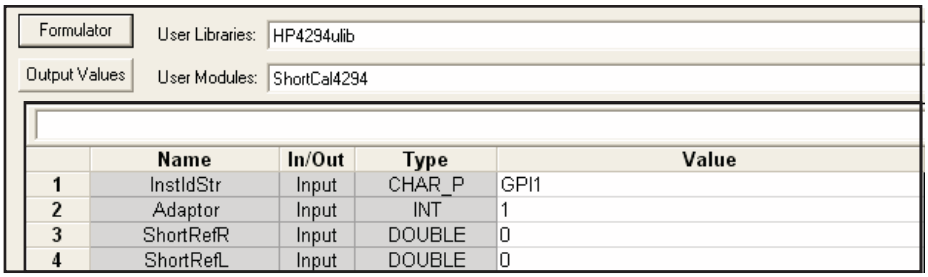
User module description

The ShortCal4294 routine performs SHORT calibration for the HP 4294 IMP meter. Figure N-6 shows the default parameters for the ShortCal4294 user module. After SHORT calibration is performed, it should then be verified by performing an FiSweep on the calibration device.

Syntax:

status = ShortCal4294(char *Inststr, int Adaptor, double ShortRefR, double ShortRefL);

Figure N-6
ShortCal4294 user module (default parameters)



INPUTS:

InstIdStr (char *) KCON instrument ID. Default is CMTR1.

Adaptor (int) Type of cable adapter. Valid input is 1 to 4:

1 Specifies Agilent 16048G (2 pairs of 1-meter coaxial cables).
2 Specifies Agilent 16048H (2 pairs of 2-meter coaxial cables).
3 Specifies Agilent 42942A (7-millimeter 42942A probe).
4 Specifies Agilent 42941A (Advanced impedance probe).

ShortRefR (double) Reference resistance value of the SHORT.

ShortRefL (double) Reference inductance value of the SHORT.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0 OK.

-**10030** HP 4294 not in KCON.

-10090 GPIB time-out occurred during communications.

-10100 An invalid input parameter is specified.

-**10102** There is an error when parsing the HP 4294's response.

wlrlib user library reference

The user modules in the wlrlib user library are used to perform linear regression and QBD ramp tests for WLR testing. These user modules are summarized in Table N-2. Details for each of the user modules follow the table.

Table N-2
WLR user modules

User module	Description
llsq1	Performs simple linear regression.
qbd_rmpv	Performs Charge-to-Breakdown test using the QBD V-ramp test.
qbd_rmpj	Performs Charge-to-Breakdown test using the QBD J-ramp test.

llsq1 user module

User module description

The llsq1 routine performs simple linear regression. This user module is not intended to be used as a stand-alone module. It is a function that is used by other modules. The llsq1 user module is shown in Figure N-7.

NOTE: *Reference: See page 175 of Snedecor, "Statistical Methods."*

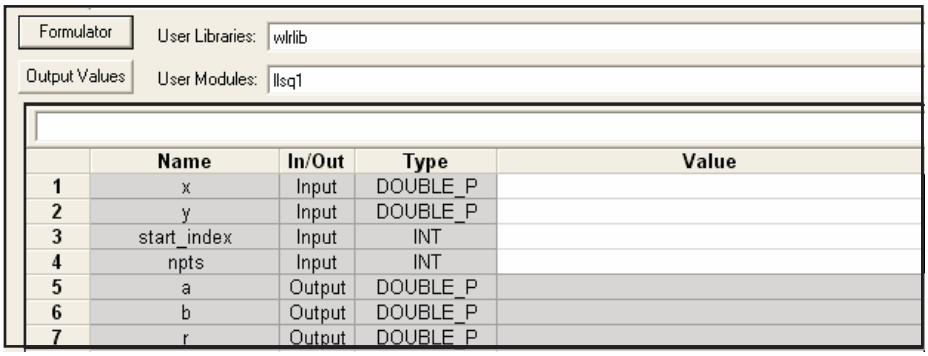
NOTE: *LLSQ will return 0.0 for a, b, and r if the slope correlation calculations have 0.0 denominators.*

CAUTION **Occasionally, it may be necessary to modify this routine. If this is done, be sure to change the name of the function. Other routines may be relying on this function. If the name is not changed, it will be overwritten in your next update.**

Syntax:

status = llsq1(double *x, double *y, int start_index, int npts, double *a, double *b, double *r);

Figure N-7
llsq1 user module



	Name	In/Out	Type	Value
1	x	Input	DOUBLE_P	
2	y	Input	DOUBLE_P	
3	start_index	Input	INT	
4	npts	Input	INT	
5	a	Output	DOUBLE_P	
6	b	Output	DOUBLE_P	
7	r	Output	DOUBLE_P	

INPUTS:

x (double *) Array of x-data.

y (double *) Array of y-data.

start_index (int) Starting index value.

npts (int) Number of data points.

OUTPUTS:

a (double *) Calculated slope.

b (double *) Calculated y-intercept.

r (double *) Correlation coefficient.

qbd_rmpv and qbd_rmpj modules

User module descriptions

qbd_rmpv This routing performs a Charge-to-Breakdown test using the QBD V-ramp test algorithm described in JESD35-A Procedure for Wafer Level Testing of Thin Dielectrics." This algorithm forces a linear voltage ramp until the oxide layer breaks down.

qbd_rmp This routine performs a Charge-to-Breakdown test using the QBD J-ramp test algorithm described in JESD35-A "Procedure for Wafer Level Testing of Thin Dielectrics." This algorithm forces a logarithmic current ramp until the oxide layer breaks down.

NOTE: *These user modules are documented in Appendix L, WLR Testing. For details, see V-ramp test: qbd_rmpv User Module and J-ramp test: qbd_rmpj User Module in Appendix M.*

Hotchuck_Triotek user library reference

The user module in the Hotchuck_Triotek user library is used to control the temperature of the Triotek hot chuck. This user module is summarized in Table N-3. Details for the user module follows the table.

Table N-3
Hotchuck_Triotek user module

User module	Description
SetChuckTemp	Sets the temperature of the Triotek hot chuck.

SetChuckTemp user module

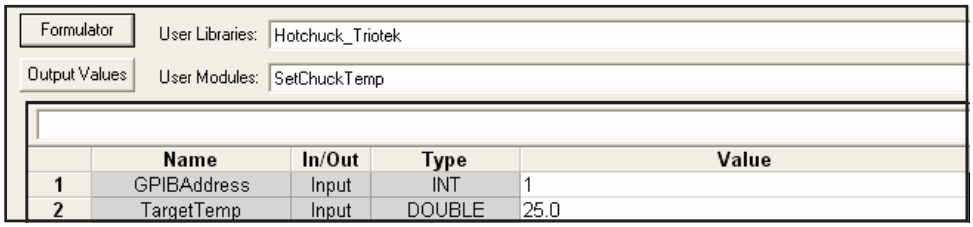
User module description

The SetChuckTemp routine is used to configure the TrioTech hot chuck to reach a desired temperature. The temperature controller that controls that hot chuck is TC1000. The SetChuckTemp user module is shown in Figure N-8.

Syntax:

status = HotChuck Temp(int GPIBAddress, double TargetTemp);

Figure N-8
SetChuckTemp user module (default parameters)



INPUTS:

TargetTemp (double) Target temperature (in ºC) for the chuck.

RETURNED STATUS VALUES:

Returned values are placed in the spread sheet (**Sheet** tab).

0 OK.

-**2** Unable to set temperature.

-**10000** (INVAL_INST_ID) The specified instrument ID does not exist.

-**10090** (GPIB_ERROR_OCCURRED) A GPIB communications error occurred.

-**10091** (GPIB_TIMEOUT) A time-out occurred during communications.

Triotek receive commands:

All receive commands have character R as the leading character. "R" commands are used with a user provided computer with IEEE for remote control of the TC controller. Below is a list of receive commands and their descriptions:

RAn Command Used to set beginning ramp number.

n = 1 to 8

RBn Command Used to activate a preset temperature.

n = 1 to 8

RDxx.x Command Used to set Dewpoint Limit.

$-20.0 \leq xx.x \leq 20.0$

RSxxx.x Command Used to set temperature.

Lower Temp Limit $\leq$ xxx.x $\leq$ Upper Temp Limit

RPn Command Used to inhibit chuck power.

n = 0 power inhibited

n = 1 power enabled

RNn Command Used to set program cycle.

$1 \leq n \leq 999$

RGn Command Used to toggle program mode.

n = 0 program halt

n = 1 program restart

RXn Command Used to enable program mode.

n = 0 program mode disabled

n = 1 program mode enabled

RnLxxx.x Command Used to set preset #n to temperature xxx.x.

n = 1 to 8

Lower temp limit $\leq$ xxx.x $\leq$ upper temp limit

RnSxxx.x Command Used to set program temp. Set point #n to xxx.x.

n = 1 to 8

Lower temp limit ≤ xxx.x ≤ upper temp limit

RnWx.x Command Used to set program window #n to x.x.

n = 1 to 8

0.1 ≤ x.x ≤ 9.9

RnRm Command Used to set program ramp time #n to m seconds.

n = 1 to 8

0 ≤ m ≤ 9999 seconds

RnKm Command Used to set program dwell time #n to m seconds.

n = 1 to 8

0 ≤ m ≤ 9999 seconds

Triotek Send Commands:

All send commands have S as the leading character. All responses for send commands do not include headers. "S" commands are used with a user provided computer with IEEE for remote monitoring of the TC controller. Below is a list of send commands and their descriptions:

ST Command Used to query current chuck temperature.

SD Command Used to query current Dewpoint.

SL Command Used to query set Dewpoint limit.

SM Command Used to query Service Request byte. Refer to Service Request byte for return format.

SF Command Used to query Condition Flag byte. Refer to Condition Flag byte description for return format.

SS Command Used to query local set temperature.

SW Command Used to query set window value.

SA Command Used to query program start step.

SN Command Used to query program cycle count.

SH Command Used to query the program status byte.

SI Command Used to query program cycle remaining.

SE Command Used to query over heat.

SnL Command Used to query set point for preset #n.

n = 1 to 8

SnS Command Used to query set temperature for program set point #n.

n = 1 to 8

SnW Command Used to query program set #n window value.

n = 1 to 8

SnR Command Used to query program set #n ramp time.

n = 1 to 8

SnK Command Used to query program set #n dwell time.

n = 1 to 8

Appendix O
Advanced Applications

In this section:

Advanced Applications

In this section, you will learn the following:

- **Controlling a switch matrix** — Demonstrates how to use a switch matrix to automatically connect any instrument terminal to any test system pin.
- **Sequencing tests on multiple devices** — Demonstrates how to run a test sequence that will utilize a switch matrix to automatically test all of the devices in the ivswitch project.
- **Customizing a user test module (UTM)** — Demonstrates how to modify a user module using the Keithley User Library Tool (KULT).

The following equipment is required to complete this tutorial and obtain data that functionally correlates with data included with the ivswitch sample project.

- 1 - Keithley Model 4200-SCS with a total of three SMUs (Preamps not required)
- 1 - Keithley Model 8006 Component Test Fixture
- 1 - Keithley Model 707A or 708 Switch Matrix
- 1 - Keithley Model 7072 or 7174A 8×12 matrix card
- 4 - Keithley Model 4200-MTRX-X cables (0 if using preamps)
- 12 - Keithley Model 4200-TRX-X cables (16 if using preamps)
- 1 - Keithley Model 7007 GPIB cable
- 1 - Keithley Model 236-ILC-3 safety interlock cable
- 1 - NPN transistor (2N3904 or similar)
- 1 - N-channel MOSFET (Temic SD210DE or similar)
- 1 - Capacitor (10 pF)
- 1 - Resistor (1 G Ω)
- 1 - Diode (1N970B or similar)

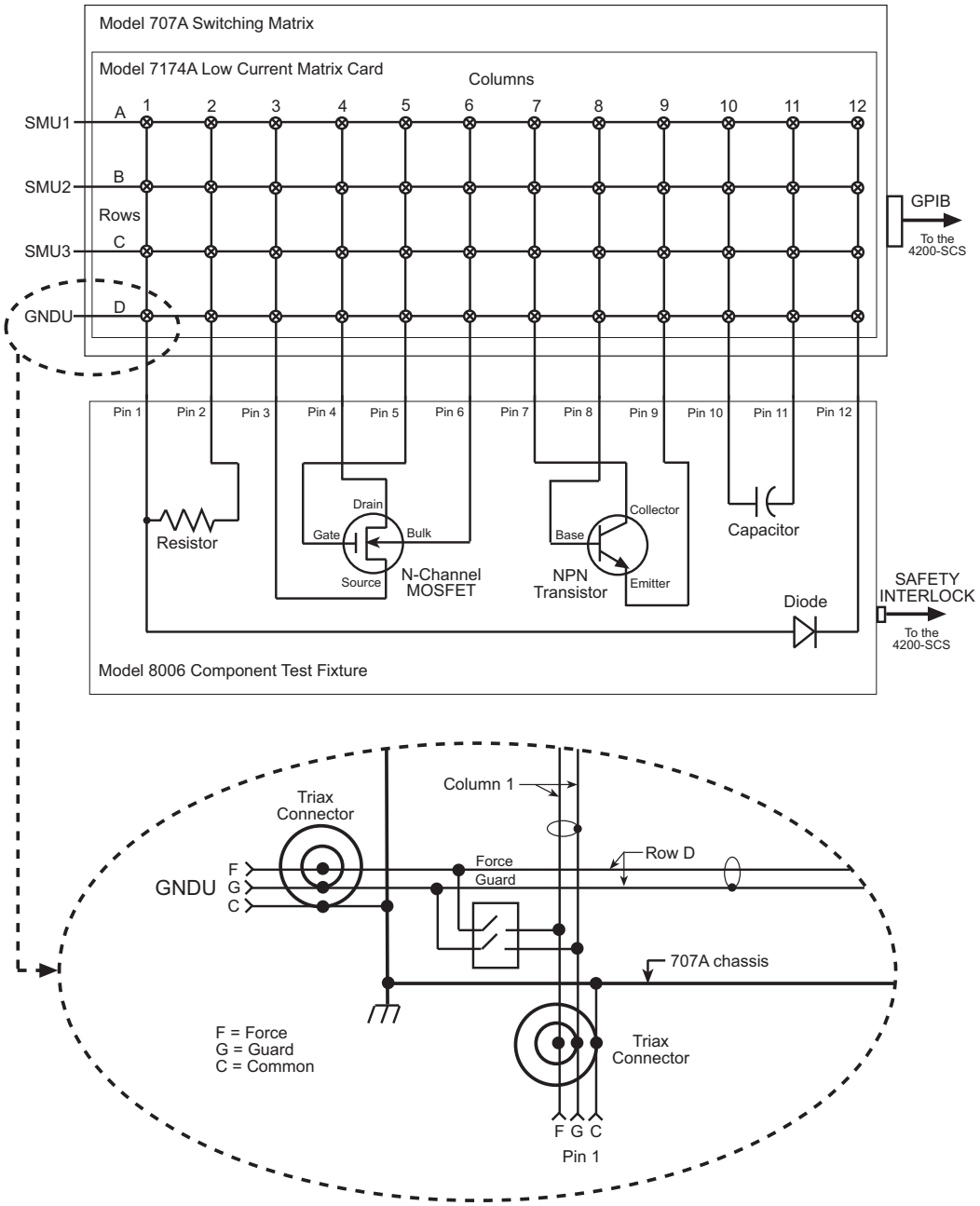
Controlling a switch matrix

This tutorial demonstrates how to use a switch matrix to automatically connect any instrument terminal to any test system pin. The ivswitch sample project will be used to illustrate this functionality. Before loading and running the ivswitch project, the 4200-SCS, switch matrix, and component test fixture must be connected as illustrated in Figure O-1.

The switch matrix is controlled by the 4200-SCS via the General Purpose Instrument Bus (GPIB) bus. Use a Model 7007 GPIB cable to connect the Model 707 Switching Matrix to the 4200-SCS. For connection details, refer to GPIB connections in Appendix B. This example shows a Model 7174A matrix card installed in slot 1 of a Model 707A Switching Matrix. The row-column connection scheme is used for this tutorial.

A UTM is used to control the switch matrix. When a test sequence for a device is started, the UTM will close the appropriate matrix crosspoints to connect the specified instrument terminals to the appropriate test system pins. For details on UTMs, refer to the 4200-SCS Reference manual.

Figure O-1
Devices connected to 707A switching matrix



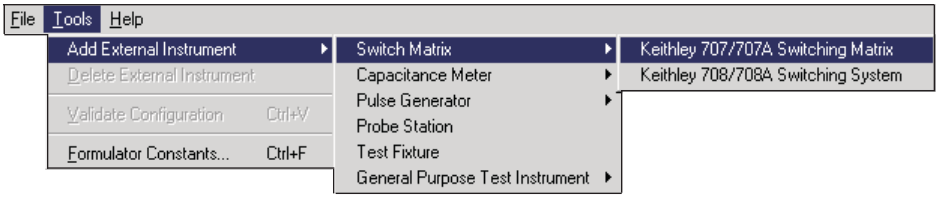
KCON setup

After connecting the system as indicated in Figure O-1, run the Keithley CONfiguration utility (KCON) to add the switch matrix and test fixture to the system configuration. KCON is used to manage the configuration of all instrumentation controlled by the 4200-SCS software. Once the matrix and test fixture have been added, and the instrument-to-matrix-to-pins connections have been defined, specify an instrument terminal and test system pin and KITE will automatically connect the two using the matrix. Changes to the system configuration will only be necessary when changes to instrument-to-matrix-to-pins wiring are made.

Follow the steps below to start KCON and modify the system configuration as described above. For additional information regarding KCON, refer to the Keithley CONfiguration Utility (KCON), Section 7.

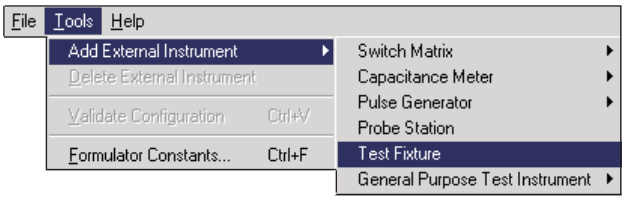
1. On the desktop, double-click the **KCON** icon to open KCON.
2. Using the **Tools** menu, add a switch matrix to the system configuration as indicated in Figure O-2.

Figure O-2
Add a switch matrix to the system configuration



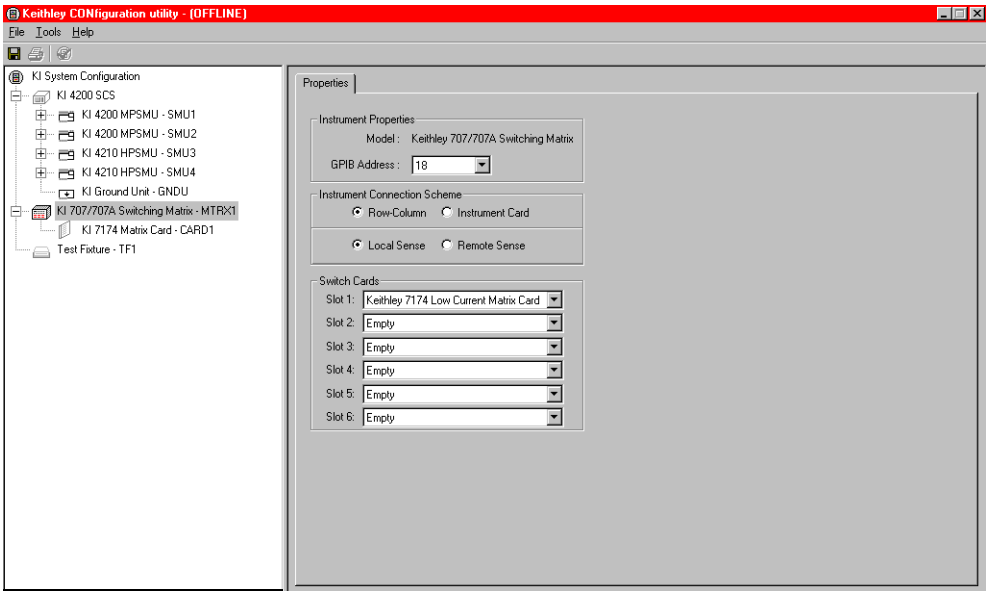
3. Using the **Tools** menu, add a test fixture to the system configuration as indicated in Figure O-3.

Figure O-3
Add a test fixture to the system configuration



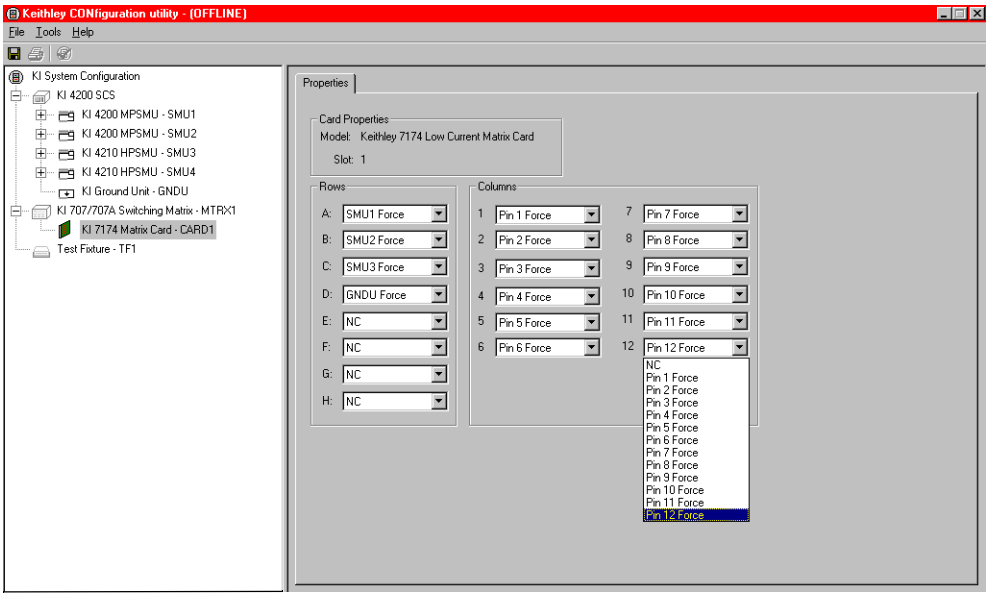
4. Select the **KI 707/707A Switching Matrix - MTRX1** item in the Configuration Navigator (tree control on left side of screen) and add a **Keithley 7174 Low Current Matrix Card** to **Slot 1** of the switch matrix. Add the switch card using the pull down menu on the **Properties** tab. See Figure O-4.

Figure O-4
Add a switch card to the system configuration



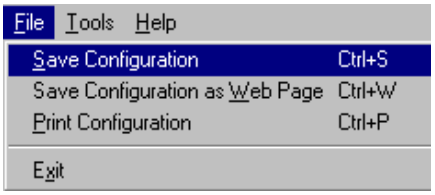
5. Select the **KI 7174 Matrix Card - CARD1** item in the Configuration Navigator. Connect the SMUs, GNDU, and test fixture pins as indicated in Figure O-1 using the pull down menus on the **Properties** tab. See Figure O-5.

Figure O-5
Define the system connections



6. **Save** the system configuration and **Exit** KCON. See Figure O-6.

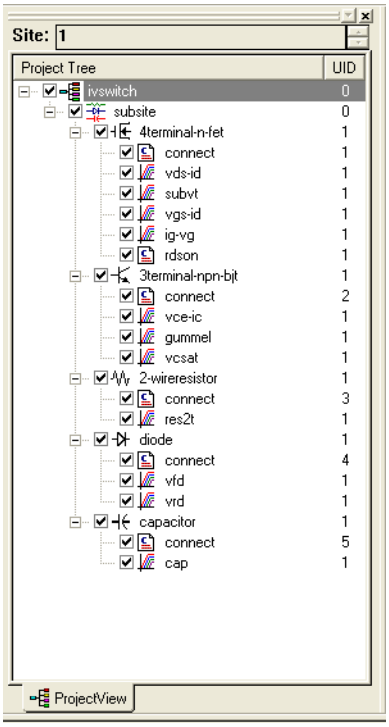
Figure O-6
Save the system configuration



Open KITE and the "ivswitch" project

1. On the desktop, double-click the **KITE** icon to open KITE.
2. Open the **ivswitch** project from the **File** menu on the KITE toolbar (click **Open Project**). The Project Navigator for the ivswitch project is shown in Figure O-7.

Figure O-7
Project Navigator - "ivswitch" project



Running test sequences

NOTE *For detailed information regarding test and sequence execution, refer to the Run execution of individual tests and test sequences in Section 6 of the Reference Manual.*

The ivswitch project uses the same Interactive Test Modules (ITMs) that are used in the default project. The primary difference between the two projects is that the ivswitch project uses connect UTMs to control the switch matrix. As shown in Figure O-7, there is a connect UTM at the beginning of each device test sequence.

A test sequence for a device is executed by selecting the **Device Plan**, and then clicking the green **Run** button. When a Device Plan is started, the connect test closes the appropriate matrix crosspoints to connect the instruments to the appropriate device.

All devices may be tested by selecting the **Subsite Plan** and clicking the green **Run** button.

Figure O-8 through Figure O-12 show the signal paths that are automatically selected for the five devices.

Figure O-8
Signal paths for "4terminal-n-fet" tests

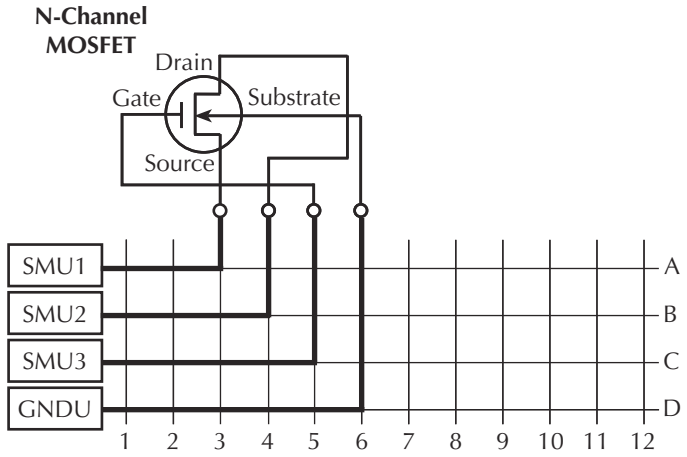


Figure O-9
Signal paths for "3terminal-npn-bjt" tests

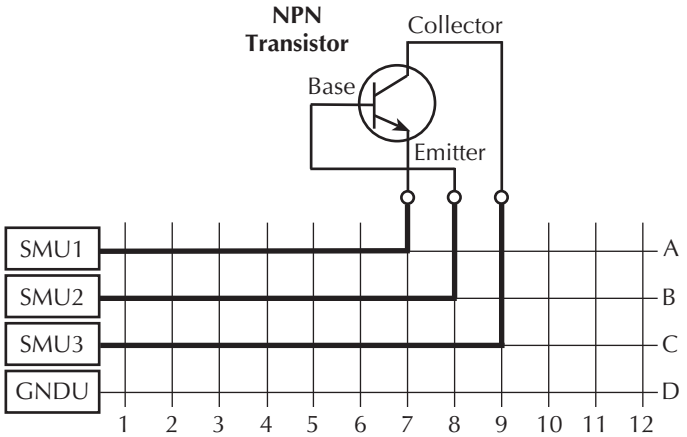


Figure O-10
Signal paths for "2-wireresistor" tests

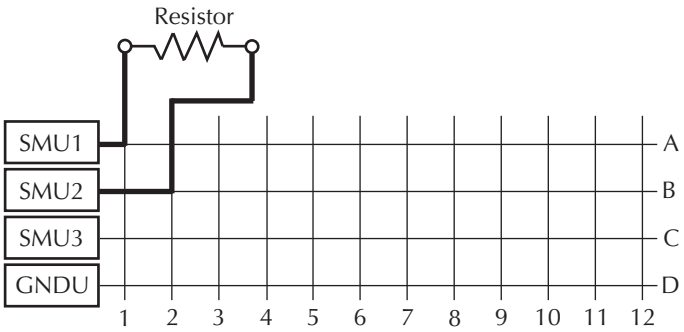


Figure O-11
Signal paths for "diode" tests

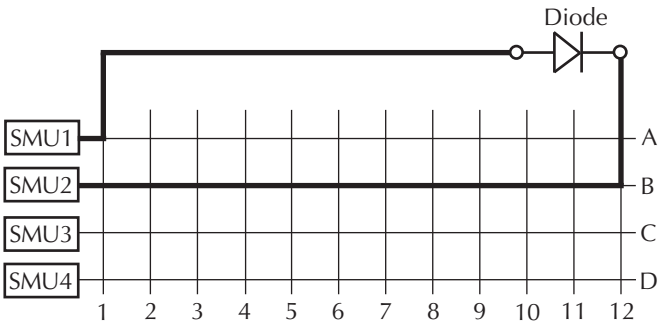
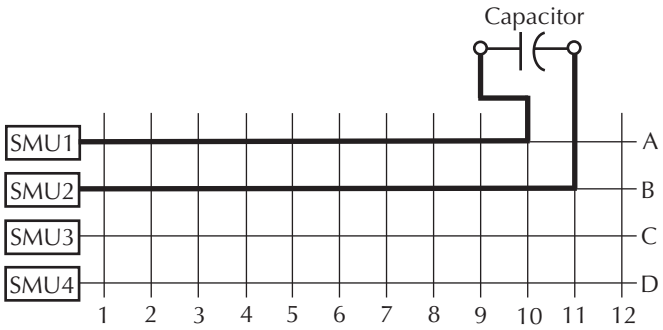


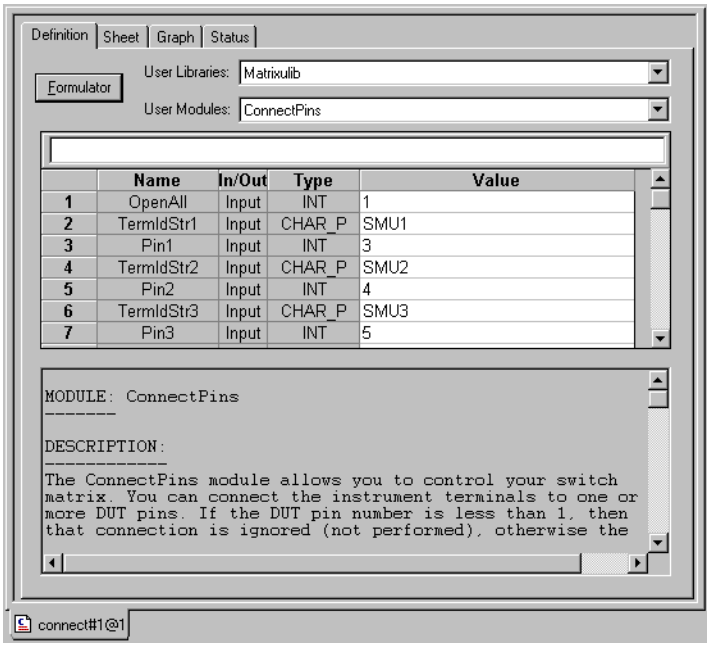
Figure O-12
Signal paths for "capacitor" test



"connect" test description

In the Project Navigator, double-click **connect** under the 4terminal-n-fet device to open the test. The test is shown in Figure O-13.

Figure O-13
"connect" test



The connect test is a User Test Module (UTM). KITE supports two types of test modules; ITMs and UTMs. A UTM, like an ITM, has **Definition**, **Sheet**, **Graph**, and **Status** tabs. The operation of each tab, regardless of test module type, is identical except for the **Definition** tab.

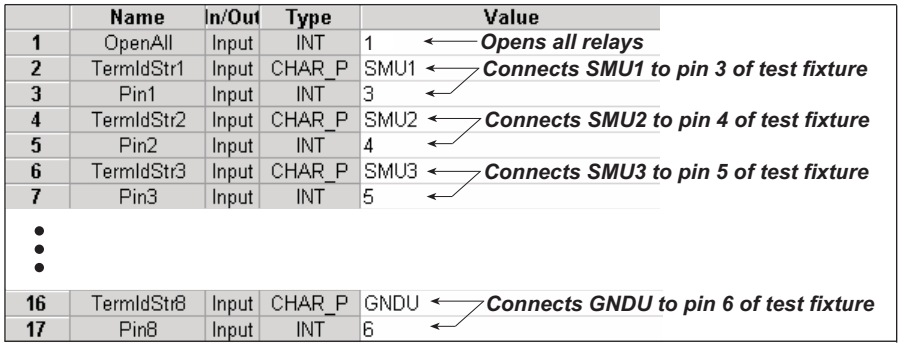
On the UTM **Definition** tab, the user connects the UTM to a user module located within a User Library, and sets the module parameter values. This information is stored with the UTM when it is saved. When a UTM is executed, the parameters will be passed from the UTM to the user module and the user module will be executed. User libraries and user modules are created and managed using KULT. Refer to the Keithley User Library Tool (KULT), Section 8 for more information regarding user libraries.

In this example, the connect UTM is connected to the **ConnectPins** user module in the **Matrixulib** user library. **ConnectPins** has a total of 17 parameters. The first parameter, **OpenAll**, will cause **ConnectPins** to open all matrix crosspoints before closing any additional crosspoints. It is a good practice to open all the switch connections before making any new closures. Inadvertent switch closures may damage DUT.

The 16 additional parameters are comprised of eight terminal-pin-pairs. As shown in Figure O-14, each specified terminal-pin-pair causes **ConnectPins** to make the desired matrix connection. Because the instrument-to-matrix-to-pin connectivity was defined using KCON, KITE is able to automatically connect the specified instrument terminals to the appropriate tester pins.

NOTE *If a Pin parameter is < 1, the terminal-pin-pair is ignored and no matrix connections are made.*

Figure O-14
"connect" parameters for "4terminal-n-fet" device



Sequencing tests on multiple devices

For the previous tutorial, a switch matrix was added to the test system to automate connection changes for different devices. When a test sequence for a device (Device Plan) is executed, the connect test closes the appropriate matrix crosspoints to connect that device to the appropriate instrumentation. The test sequence stops after the Device Plan has been executed.

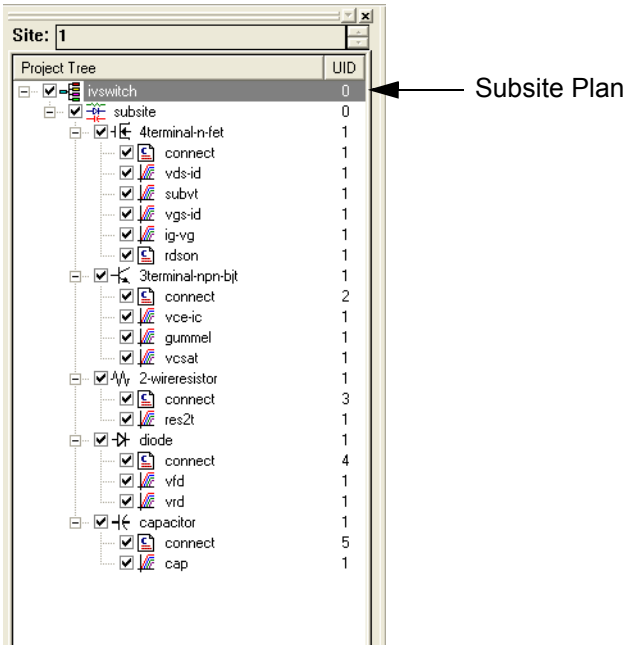
This tutorial demonstrates how to run a test sequence that will automatically test all the devices in the ivswitch project. After all the devices have been tested, the test sequence will stop.

Open "ivswitch" project

If the ivswitch project is not currently open, open it using the **Open Project** item of the **File** menu on the toolbar. The Project Navigator for the ivswitch project is shown in Figure O-15.

With a switch matrix added to the system, all the devices can be tested by starting the test sequence from the subsite level of the Project Navigator. Figure O-15 shows the **Subsite Plan** selected (highlighted) to execute.

Figure O-15
Project Navigator - "ivswitch" project



Modify test sequence

The Project Navigator shows the execution sequence for the **Subsite Plan**. As shown in Figure O-15, the 4terminal-n-fet will be tested first, followed by tests for the other four devices.

The device test sequence may be changed using the **Subsite Plan Sequence** tab. The following exercise shows how to change the test sequence by making diode the first device in the sequence:

1. In the Project Navigator, double-click **"subsite"** to open the **Subsite Plan** window (see Figure O-16).
2. In the **Device Sequence Table**, click **diode"** to select it. Figure O-16 shows diode selected.
3. Use the **Move Up** button to move diode to the top of the sequence table (Figure O-17).
4. At the bottom right-hand corner of the **Subsite Plan** window, click the **Apply** button to change the sequence (see Figure O-18).

Figure O-16
Subsite Plan window

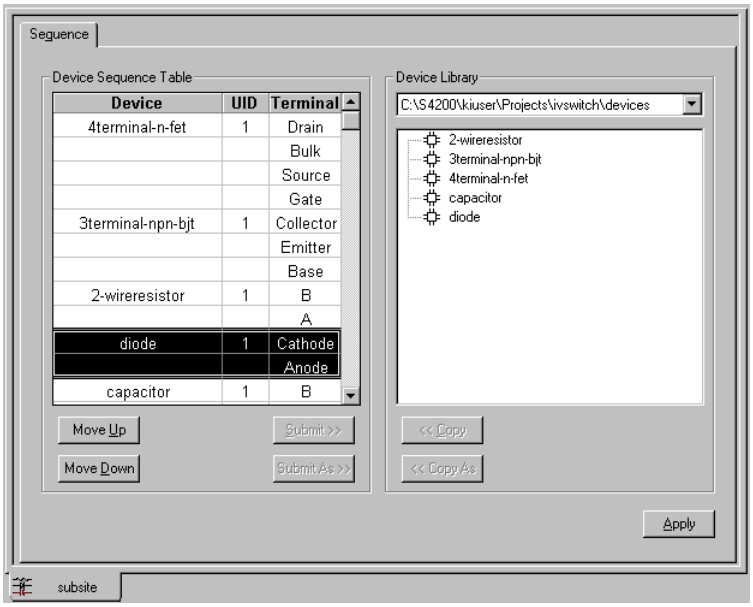


Figure O-17
"diode" moved to top of sequence table

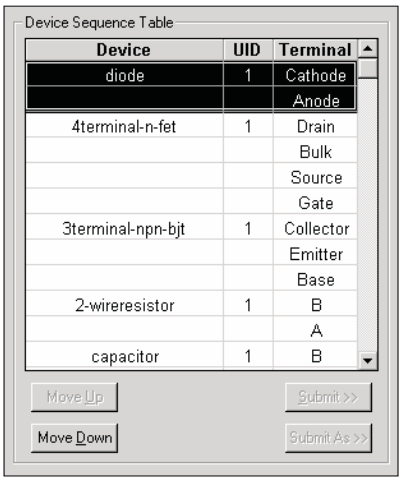
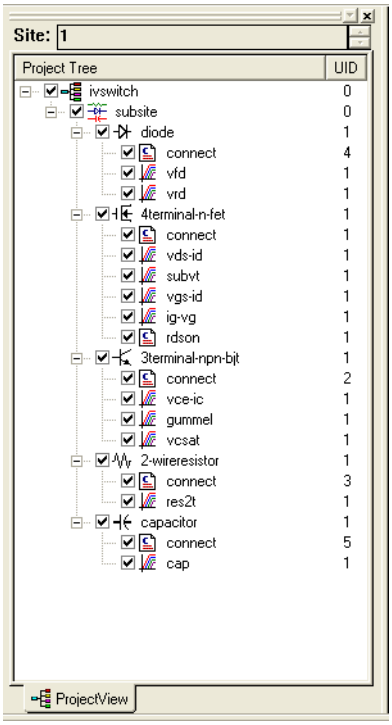


Figure O-18
"diode" moved to top of Project Navigator



Execute the test sequence (Subsite Plan)

To select the **Subsite Plan**, click **"subsite"** in the Project Navigator. The **Subsite Plan** name will appear in the execution indicator box as shown in Figure O-19.

To execute the **Subsite Plan**, click the green **Run** button. The first test for each device will control the switch matrix, which connects the device to the instrumentation. The switch matrix was added in the previous application Controlling a switch matrix on page O-2.

While each test is running, the test name will appear in the execution indicator box. After the last test vt is executed, the testing process will stop.

Figure O-19
Execution indicator box

Customizing a user test module (UTM)

This tutorial demonstrates how to modify a user module using KULT. In the ivswitch project, there is a test named rdson. The rdson test measures the drain-to-source resistance of a saturated N-channel MOSFET as follows:

1. Applies 2 V to the gate (Vg) to saturate the MOSFET.
2. Applies 3 V to the drain (Vd1) and performs a current measurement (Id1).
3. Applies 5 V to the drain (Vd2) and performs another current measurement (Id2).
4. Calculates the drain-to-source resistance rdson as follows:

rdson = (Vd2-Vd1) / (Id2-Id1)

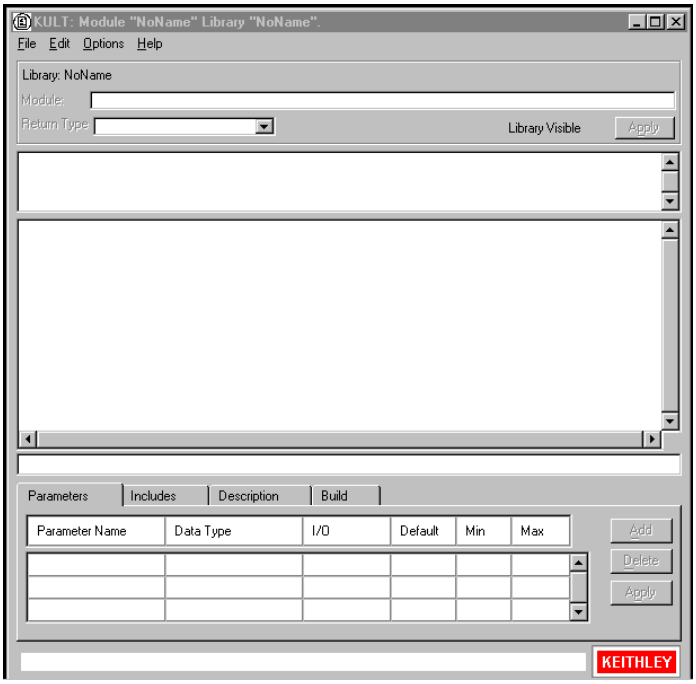
The rdson test has a potential shortcoming. If the drain current is noisy, the two current measurements may not be representative of the actual drain current. Therefore, the calculated resistance may be incorrect.

In this example, the user module will be modified in KULT such that 10 current measurements will be performed at Vd1 and 10 more at Vd2. The current readings at Vd1 will be averaged to yield Id1, and the current readings at Vd2 will be averaged to yield Id2. Using averaged current readings smooths out the noise. For details on using KULT, refer to the Keithley User Library Tool (KULT), Section 8.

Open KULT

From the desktop, open the KULT tool by double-clicking the **KULT** icon. The **KULT** main window is shown in Figure O-20.

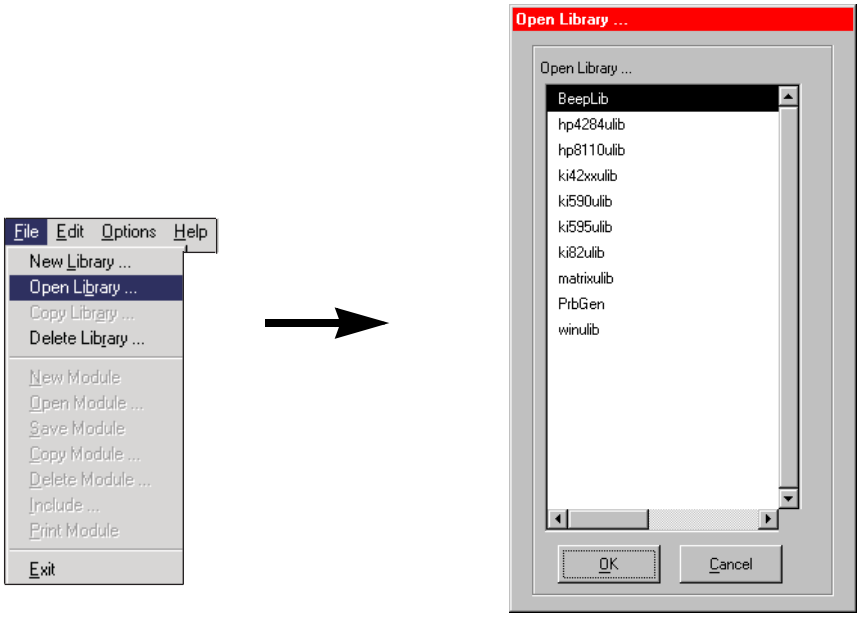
Figure O-20
KULT main window



Open the "ki42xxulib" user library

1. From the **File** menu, select the **Open Library** item (see Figure O-21A).
2. From the **Open Library** window, select **ki42xxulib** as shown in Figure O-21B and click **OK**.

Figure O-21
Open "ki42xxulib" library



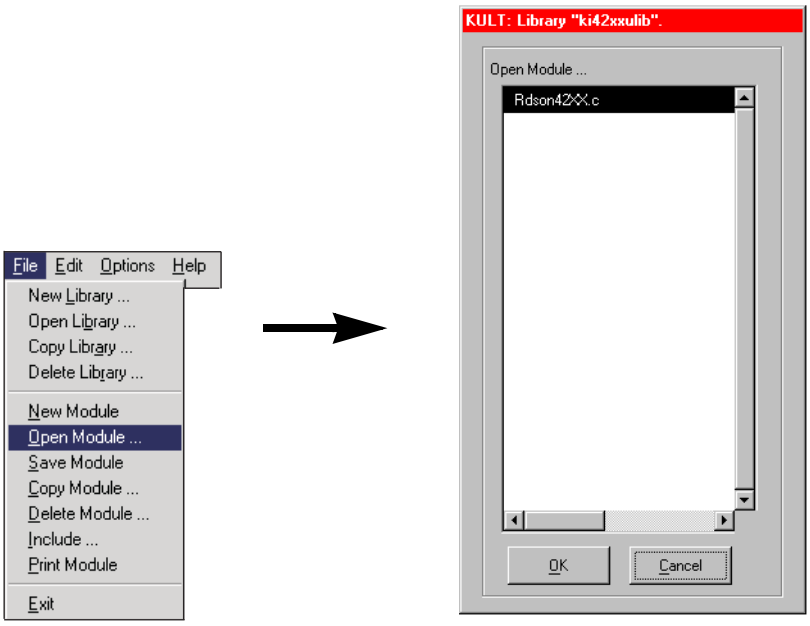
A. Select **Open Library**

B. Select "**ki42xxulib**" and click **OK**

Open the "Rdson42XX" user module

1. From the **File** menu, select the **Open Module** item (see Figure O-22A).
2. From the **Open Module** window, select **Rdson42XX.c** as shown in Figure O-22B, and click **OK**. The **Rdson42XX** module will open.

Figure O-22
Open "Rdson42XX" module



A. Select **Open Module**

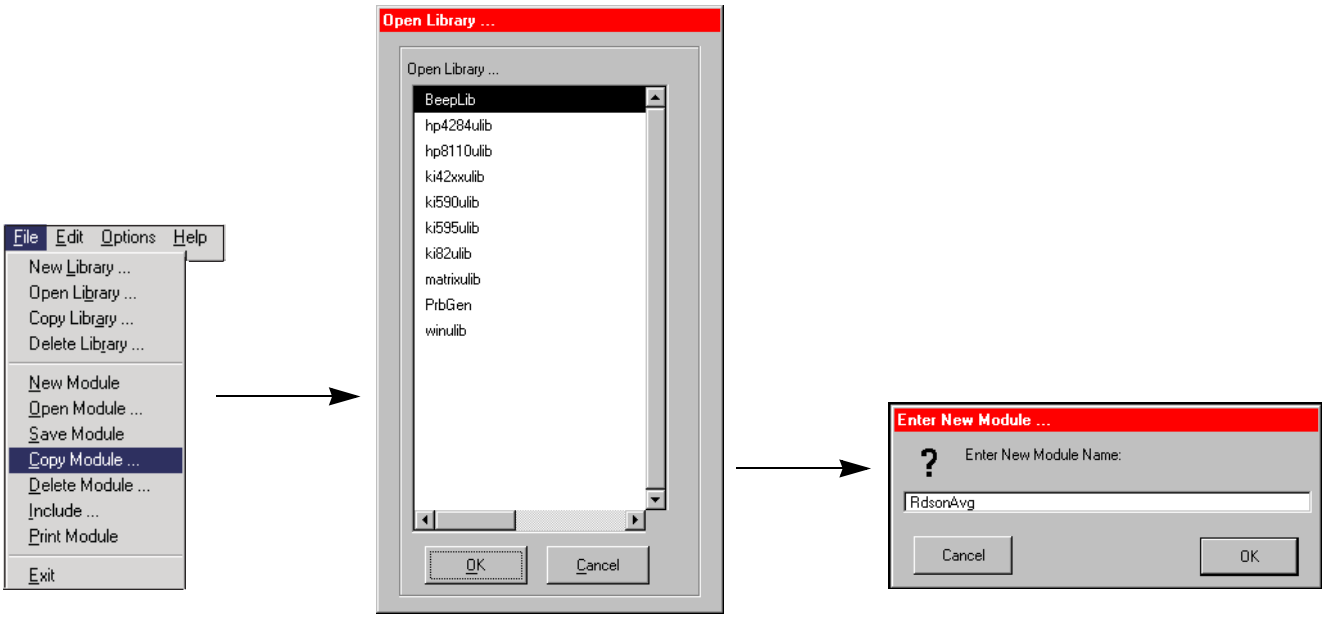
B. Select "**Rdson42XX.c**" and click **OK**

Copy "Rdson42XX" to "RdsonAvg"

The new module will be created by copying the Rdson42XX module as RdsonAvg and then making the appropriate changes to the test module.

1. From the **File** menu, select the **Copy Module** item (see Figure O-23A).
2. From the **Copy Module** window, select **ki42xxulib** as shown in Figure O-23B and click **OK**. This selects the library for the module.
3. From the **Enter New Module Name** window, type in the name as shown in Figure O-23C and click **OK**. A dialog box will remind you that the library using the new module will have to be built. Click **OK**.

Figure O-23
Copy "Rdson42xx" module as "RdsonAvg"

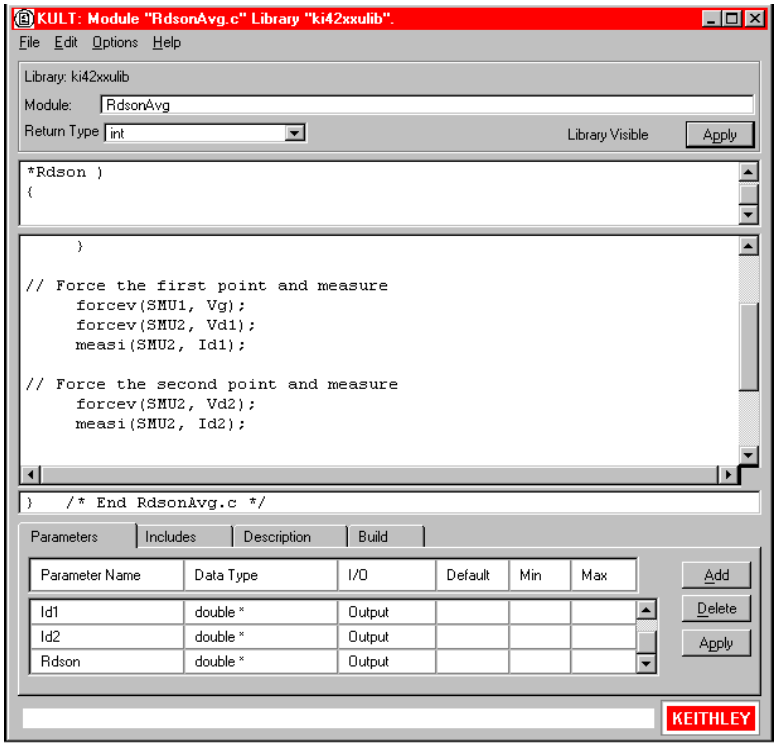


A. Select **Copy Module** B. Select "**ki42xxulib**" and click **OK** C. Type in "**RdsonAvg**" and click **OK**

Open and modify the "RdsonAvg" user module

From the **File** menu, select **Open Module**, and then select **RdsonAvg.c** from the **Open Module** window. The RdsonAvg module is shown in Figure O-24.

Figure O-24
KULT module window



Modify the user module code

The **measi** commands are to be replaced with **avgi** commands. While a **measi** command performs a single measurement, an **avgi** command performs a specified number of measurements, and then calculates the average reading. For example:

avgi (SMU2, Id1, 10, 0.01);

For the above command, **SMU2** performs 10 current measurements and then calculates the average reading (**Id1**). The **0.01** parameter is the delay between measurements (**10 m**s).

The source code for the module is located in the module code area of the window. In this area, make the changes indicated in the following note:

NOTE *For details on modifying a KULT program, refer to the* Keithley User Library Tool (KULT), Section 8*.*

Figure O-25
Program modifications

```
// Make the connections
   if ((GatePin > 0) && (DrainPin > 0) && (SourcePin > 0))
   {
      // Switch matrix used.
         conpin(SMU1, GatePin, 0);
         conpin(SMU2, DrainPin, 0);
         conpin(GND, SMU1L, SMU2L, SourcePin, BulkPin, 0);
   }

// Force the first point and measure
   forcev(SMU1, Vg);
   forcev(SMU2, Vd1);
   measi(SMU2, Id1);

// Force the second point and measure
   forcev(SMU2, Vd2);
   measi(SMU2, Id2);

// Clean up and clear the connections and the instrument
   devint();

   *Rdson = (Vd2-Vd1)/(Id2-Id1);   // Calculate Rdson

   return( OK );

// Program changes:

   *Rdson10 = (Vd2-Vd1)/(Id2-Id1);   // Calculate Rdson10
   avgi (SMU2, Id2, 10, 0.01);  // Perform averaged I measurement
   avgi (SMU2, Id1, 10, 0.01);  // Perform averaged I measurement
```

Change a parameter name

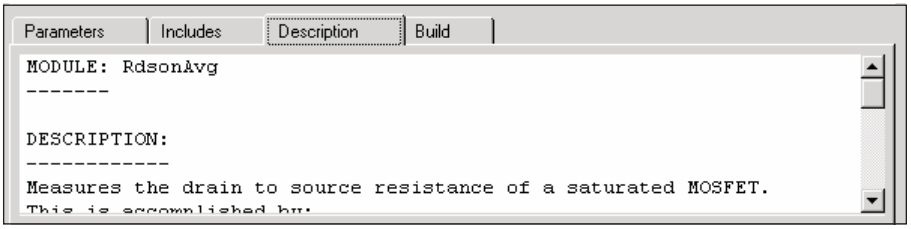
With the **Parameters** tab selected, the parameter names for the module are listed in a table located at the bottom of the window. Change the parameter name **"Rdson"** (shown in Figure O-24) to **"Rdson10"**. After typing in the new parameter name, click **Apply** to enter the change.

NOTE *For details on the Parameters tab, refer to the User's Manual, Parameters tab area, page 6-6.*

Change the module description

Click the **Description** tab to display the description for the module. Above **DESCRIPTION**, change **MODULE: Rdson42xx** to **MODULE: RdsonAvg** as shown in Figure O-26. In addition, replace all occurrences of **Rdson** with **Rdson10**. In KITE, any UTMs that are connected to this module will show the text that is entered on the **Description** tab in KULT.

Figure O-26
Module name for Description

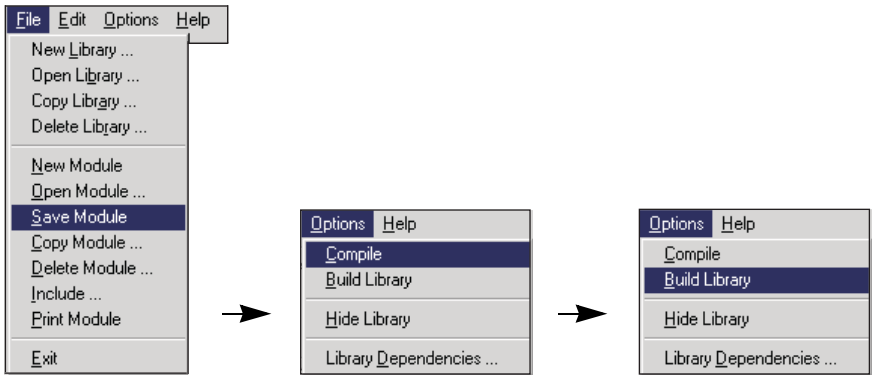


Save, compile, and build the modified library

The user module has to be saved and compiled. Finally, the library must be rebuilt to ensure that the new module is available for use by KITE UTMs. These operations are performed from the **File** and **Options** menus.

In the order shown in Figure O-27, **save**, **compile**, and **build** the library. Note that *pop-up* windows will be displayed to indicate that the compile and library building operations are in process. For details, refer to the Keithley User Library Tool (KULT), Section 8.

Figure O-27
Save, compile, and build library



A. Select **Save Module** B. Select **Compile Library** C. Select **Build**

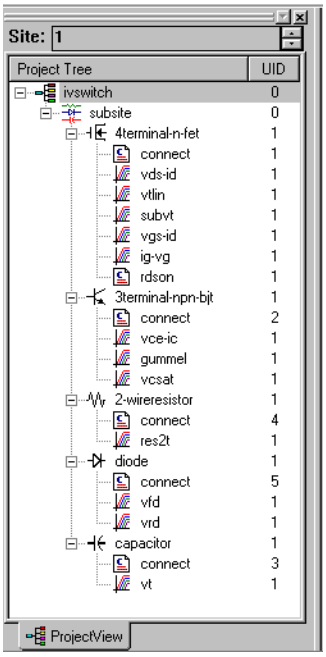
Add a new UTM to the "ivswitch" project

Open KITE and load the "ivswitch" project

1. From the desktop, open KITE by double-clicking the **KITE** icon.
2. Open the **ivswitch** project from the **File** menu.

The Project Navigator for the ivswitch project is shown in Figure O-28. Notice that **rdson** is the last test for the 4terminal-n-fet device.

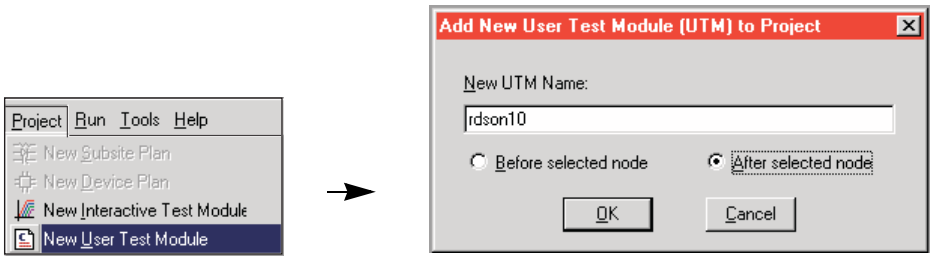
Figure O-28
Project Navigator for "ivswitch" project



Add a new UTM

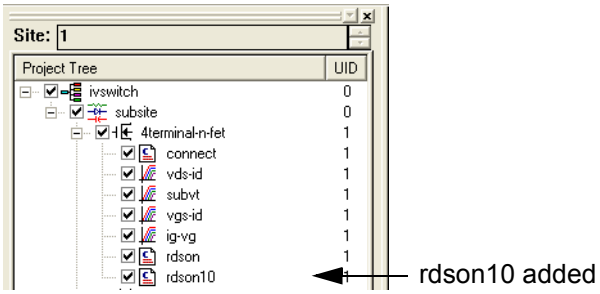
1. In the Project Navigator, single-click **rdson** to select it. This establishes the position for the new UTM.
2. From the **Project** menu, select **New User Test Module** (see Figure O-29A).
3. In the **Add New User Test Module (UTM) to Project** window, type in the new name as shown in Figure O-29B and click **OK**. Figure O-30 shows the new UTM added to the Project Navigator.

Figure O-29
Add new UTM



A. Select **New User Test Module** B. Type in **"rdson10"** and click **OK**

Figure O-30
"rdson10" added to Project Navigator

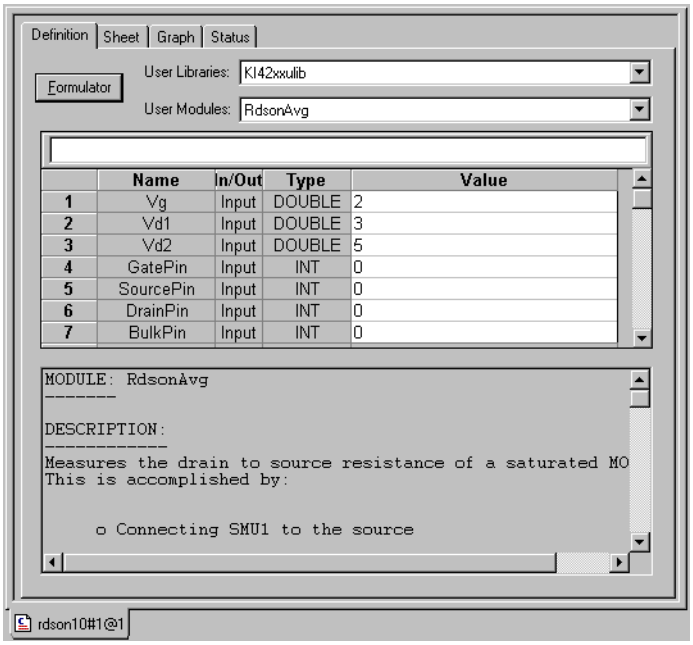


Connect the "rdson10" UTM to the "RdsonAvg" user module

In the Project Navigator, double-click **"rdson10"** test to open it. The test will open in the Workspace with the **Definition** tab blank.

Referring to Figure O-31, select **"ki42xxulib"** from the drop-down menu for the **User Libraries** box in the UTM. Next, select **"RdsonAvg"** from the menu for the **User Modules** box.

Figure O-31
"rdson10" UTM



Test description

The rdson10 test measures the drain-to-source resistance of a saturated MOSFET. Using the user-input parameter values shown in Figure O-31, the MOSFET is tested as follows when rdson10 is executed:

1. Applies **2** V to the gate (**Vg**) to saturate the MOSFET.
2. Applies **3** V to the drain (**Vd1**) and performs 10 current measurements.
3. Averages the 10 current readings to yield a single reading (Id1).
4. Applies **5** V to the drain (**Vd2**) and performs 10 more current measurements.
5. Averages the 10 current readings to yield a single reading (Id2).
6. Calculates the drain-to-source resistance (rdson10) as follows:
7. Rdson10 = (Vd2-Vd1) / (Id2-Id1)

Symbols

Numerics

A

B

C

D

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

KEITHLEY

A GREATER MEASURE OF CONFIDENCE

Keithley Instruments, Inc.

Corporate Headquarters • 28775 Aurora Road • Cleveland, Ohio 44139 • 440-248-0400 • Fax: 440-248-6168 • 1-888-KEITHLEY • www.keithley.com

12/06